UNIVERSITÄT DUISBURG ESSEN

MECHATRONIK
Universität Duisburg-Essen | www.imech.de

A Study to Implement a 2D Laser Scanner to Determine the Platform Position and Orientation of a Cable Robot for Logistic Applications

Von der Fakultät für Ingenieurwissenschaften,
Abteilung Maschinenbau und Verfahrenstechnik der
Universität Duisburg-Essen

zur Erlangung des akademischen Grades

eines

Doktors der Ingenieurwissenschaften

Dr.-Ing.

genehmigte Dissertation

von

Rudi Kurniawan
aus
Teunom, Indonesien

Referent : Prof. Dr.-Ing. Dr. h.c. Dieter Schramm
Korreferent : Prof. Dr. Ahmad Kamal Ariffin
Tag der mündlichen Prüfung:19.07.2017

Kurzfassung

Der Einsatz seilgetriebener Parallelmanipulatoren (CDPR) in der Industrie ist der Weg in eine vielversprechende Zukunft. Im Vergleich zum klassischen Parallelmanipulator hat der CDPR einen größeren Arbeitsraum und verbraucht dennoch weniger Energie. Ein CDPR-Prototyp für die Anwendung im Hochregallager wurde im Lehrstuhl für Mechatronik der Universität Duisburg-Essen entwickelt. Ziel ist, diese Anwendung auf den Markt zu bringen.

Für die Roboterkalibrierung, zur Bewertung der Roboterregelung und aus Gründen der Sicherheit muss die Position und Orientierung (Pose) der CABLAR-Plattform bestimmt werden. Aktuelle Forschungsarbeiten zeigen, dass die effektivsten Verfahren zur Bestimmung der Plattformpose die direkte Messung mit einem externen Sensor und die indirekte Messung mit einem integrierten Sensor am Seilroboter sind. Beispiele für die direkte und indirekte Messung sind das Kamerasystem und die Vorwärtskinematik. Aufgrund der hohen Kosten ist das Kamerasystem für diese Anwendung weniger geeignet. Andererseits birgt auch die Vorwärtskinematik einige Nachteile: Die aktuelle Geometrie des Roboters stimmt wahrscheinlich wegen der Herstellungs- und/oder Montagetoleranz nicht mit dem kinematischen Modell überein. Zudem beeinflussen Umweltfaktoren, wie z. B. die Temperatur, und eine lange Betriebszeit die Seileigenschaften (z. B. Elastizitätsmodul, Seildichte, Durchmesser). Diese Änderungen verringern die Genauigkeit der Plattformpositionierung.

Ein alternatives Verfahren zur Bestimmung der Plattformpose ist die Messung mittels eines 2D-Laserscanners und eines Orientierungsaufnehmers (IMU). In Kombination mit Reflektoren an der linken, rechten und spezielle Anordnung an oberen Seite des Roboterrahmens liefert der Laserscanner einzigartige Messdaten. Das Messergebnis des Laserscanners basiert auf dem Gerade-Ebene-Schnittpunkt, der mittels Gradientenprojektionsverfahren modelliert wird. Zudem wurde ein

Kompensationsalgorithmus entwickelt, um die Auswirkung des Geschwindigkeitseffekts auf das Messergebnis aufgrund der Plattformbewegung zu verringern. Der Näherungswert der normalen Parametrierung aller Reflektoren wird mittels der modifizierten Hough-Transformation geschätzt. Unter Zuhilfenahme dieses Wertes wurde das Messergebnis anhand eines zufälligen Stichprobenverfahrens (RANSAC-Algorithmus, englisch Random Sample Consensus) segmentiert. Ziel ist, die Messdaten der Reflektoren an der linken, rechten, und oberen Seite des Roboterrahmens zu trennen. Die Methode der kleinsten Quadrate (KQ-Methode) bestimmt anhand dieser segmentierten Messdaten den besten Wert der normalen Parametrierung jeder Geraden, die zu allen Reflektoren gehört. Aus diesen Werten werden die y- und z-Komponente und der Rollwinkel der Plattformpose bestimmt. Um die Messfähigkeit des 2D-Laserscanners vom zwei dimensionalen zum räumlichen Messen zu erweitern, wurde ein mathematisches Modell mittels einer speziellen Reflektoranordnung entwickelt. Ziel ist die Bestimmung der x-Komponente und des Gierwinkels der Plattformpose. Der Nickwinkel wird vom IMU gemessen.

Die Simulation der Plattform wurde in Ruhe und in Bewegung durchgeführt. Die Simulationsergebnisse sind als Empfehlung für den Versuch am Prototyp zu sehen. Vor dem Versuch wurde die passende Sensorschnittstelle gewählt und getestet. Zudem erfolgte die Gestaltung des Sensorkonzepts zur Datenübertragung. Der Treiber für die Sensoren und die Software für die Datenbearbeitung wurden vorbereitet und das vorgeschlagene Verfahren zur Bestimmung der Plattformpose am Prototyp getestet. Im Versuch wurde die translatorische Komponente der Plattformpose mit direkter Messung der Plattformposition validiert. Der Vergleich zeigt, dass die gemessene Plattformpose nicht an gewünschter Stelle liegt. Danach wurde das vorgeschlagene Verfahren zur Bestimmung der Plattformpose während niedriger und hoher Plattformgeschwindigkeit getestet. Das Ergebnis zeigt, dass dieses Verfahren zur Bestimmung der aktuellen Plattformpose geeignet ist.

Die oben beschriebenen Forschungsergebnisse zeigen, dass vorgeschlagene Messverfahren sich zur Bestimmung der Plattformpose in Ruhe und in Bewegung eignet. Aufgrund des günstigen Preises ist das vorgeschlagene Messsystem eine vielsprechende Möglichkeit, die in der kommerziellen Anwendung des CABLAR eingesetzt werden kann.

Abstract

The implementation of a Cable Driven Parallel Robot (CDPR) as a commercial product has a promising future due to its energy efficiency and larger workspace compared to conventional parallel manipulators. A prototype of a CDPR for warehouse applications called CABLAR has been developed at the Chair of Mechatronics at the University of Duisburg-Essen (UDE), which aims to develop the CDPR as a commercial product.

In order to benchmark the controller and calibrate the robot, for safety reasons, the platform position and orientation (pose) of the CABLAR must be measured. The current reported approaches to determine the platform pose of a cable robot are direct measurement and indirect measurement. An external sensor such as a camera system or laser tracker is used in direct measurement. Meanwhile, indirect measurement is the determination of the platform pose by forward kinematics where the input is from the proprioceptive sensor. However, the camera system is not worth implementing in the commercial product due to its high cost. On the other hand, forward kinematics has drawbacks when the defined parameters are not identical to the actual parameters. The manufacturing tolerance, assembly tolerance or changing the properties and diameter of the cable because of environmental effects (e.g. temperature) and long operation time are the reasons for this and are difficult to avoid. As a result, the actual pose of the platform could deviate from the desired pose.

In this thesis, a direct measurement method by combining a 2D laser scanner with an Inertial Measurement Unit (IMU) is proposed. Several flat reflectors are fixed on the left and right of the robot frame with a special pattern design on the upper side. The laser scanner measurement result during the operation of the CABLAR is imitated based on the line-plane intersection according to the gradient projection method. A compensation algorithm aimed at reducing the velocity effect

on the measurement result due to platform motion is proposed. The rough value of normal parametrization of each reflector is estimated using the Modified Hough Transform (mHT) from the measurement result. According to the rough value of the normal parametrization, the measurement result is segmented into a dataset corresponding to the left-, right- and upper-side reflectors by Random Sample Consensus (RANSAC). The linear least squares method is applied in order to determine the fine value of the normal parametrization of all segmented data. The y-component, z-component and roll angle of the platform pose are determined from the fine normal parametrization. A mathematical model based on the reflector special pattern design is developed. The goal is to extend the limitation of the 2D laser scanner from plane measurement to become space measurement in order to obtain the x-component and the yaw angle of the platform pose. The pitch angle is measured by the IMU.

The CABLAR model is simulated to verify the proposed measurement method for the conditions where the platform is stationary and in motion. According to the simulation results, several points are concluded as the recommendations for the experiment. Before the experiment was conducted, the suitable hardware interface of the sensors was chosen and tested. The system architecture of the data transfer was designed. The software to drive the sensors and to process the measurement data was prepared. In the experiment on the prototype, the translational components of the platform pose were validated with the direct measurement. Meanwhile the rotational components obtained from the proposed method were validated with the measurement result from the IMU. The results show that the platform position deviates from the desired pose. Furthermore, the proposed platform pose measurement method is tested for the platform in low- and high-velocity motion. The results show that the proposed measurement method is able to determine the actual platform pose.

Finally, the proposed measurement method is able to determine the platform pose when stationary and in motion. The proposed measurement system is suitable for application in the commercial CABLAR due its low cost compared to the actual reported measurement system.

Contents

List of Figures

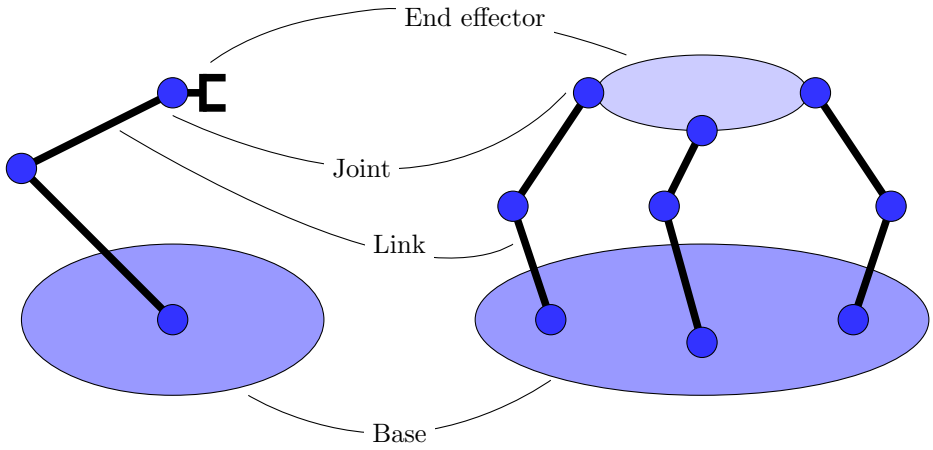
List of Tables

CHAPTER 1

Introduction

The industrial robot is used widely in industry for several purposes, in order to reduce production costs. Robots are applied either to support the labor in the production chain and/or to substitute for humans for several reasons, such as: the environment is dangerous for humans, or to shorten the time needed for production. An industrial robot mainly consists of a robot manipulator, a power supply/electrical parts, and a controller. Furthermore, the manipulator is created by arranging a base, joint, link, and end effectors with a specific topology. Two commonly used manipulator topologies in the industrial robot are serial and parallel, as depicted in Fig. 1.1.

The advantages as well as disadvantages of each topology can be found in [Pandilov and Dukovski, 2014, Verhoeven, 2004]. A comparison of their characteristics is given briefly in Table 1.1. For an application where high rigidity, low mass, and high speed are the requirement, the parallel manipulator is an appropriate choice. Meanwhile for application to a large workspace, the conventional parallel manipulator is no longer suitable due to the limited stroke of the actuator. An improvement is proposed to overcome this drawback by replacing the conventional actuator. The conventional parallel manipulator consists of three main parts: the upper plate, the actuators, and the lower plate. The upper plate (end effector) is connected by several rigid legs to the lower plate, which is the base frame of the robot. The rigid legs are the linear actuator, usually hydraulic, pneumatic or a linear electrical motor. To extend the workspace of the parallel manipulator, the rigid legs are replaced with cables. The system becomes what is referred to as a cable-based parallel manipulator –from now on, called the cable robot, for short– as depicted in Fig. 1.2.b.

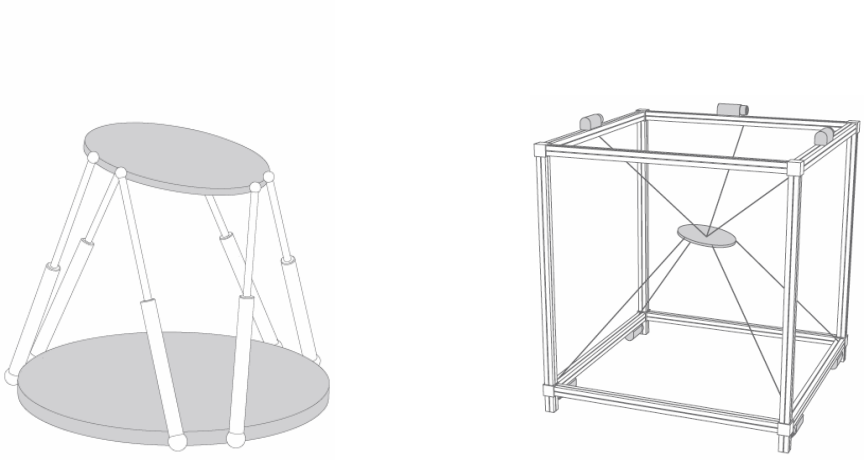


a. Serial manipulator b. Parallel manipulator

Figure 1.1: Serial and parallel manipulators.

Table 1.1: Characteristics of serial and parallel robots. Source: [Pandilov and Dukovski, 2014]

Feature	**Serial robot**	**Parallel robot**
Workspace	Large	Small & complex
Solving forward kinematics	Easy	Very difficult
Position error	Accumulates	Averages
Force error	Averages	Accumulates
Maximum force	Limited by maximum actuator force	Summation of all actuator forces
Stiffness	Low	High
Dynamical characteristics	Poor, especially with increasing size	Very high
Modeling and solving dynamics	Relatively simple	Very complex
Inertia	Large	Small
Areas of application	A great number in different areas, especially in industry	Currently limited, especially in industry
Payload/weight ratio	Low	High
Speed and acceleration	Low	High
Accuracy	Low	High
Uniformity of components	Low	High
Calibration	Relatively simple	Complicated
Workspace/robot size ratio	High	Low



a. Conventional parallel manipulator b. Cable-based parallel manipulator

Figure 1.2: Parallel Manipulator [Bruckmann et al., 2008].

The main parts of the cable robot are the end effector, the cable (metal or non-metal), a mechanism to drive the end effector, and the force sensors. The cable connects the end effector through one or more pulleys (depending on the robot's design) to the cable winch. The cable tensions are measured by the force sensors, fixed either between the cable and the end effector or between the pulley and the cable winch.

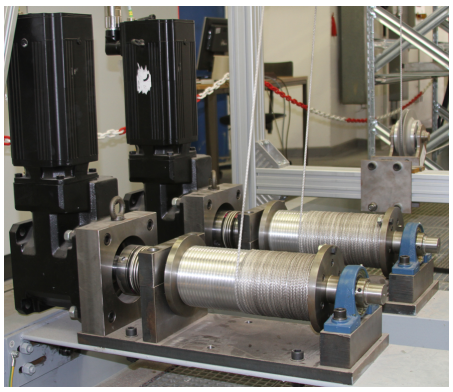
Two common mechanisms to drive an end effector are [Bruckmann et al., 2011]:

1. A winch-based system, as depicted in Fig. 1.3.a. The cable length is varied by rolling up the cable on the cable winch. This is the usual system employed on many cable robot prototypes, several of which are mentioned in [dit Sandretto et al., 2013, Miermeister et al., 2015, Verhoeven, 2004].

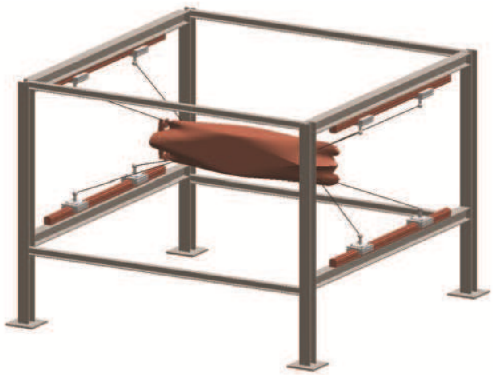
2. A rail-based system, as shown in Fig. 1.3.b. The fixed length cable connects the end effector with a skid-rail system. The posture of the end effector is controlled by changing the position of the cable anchor along the rail system.

One of the current largest cable robots in the world is a giant telescope in China, named the Five-hundred-meter Aperture Spherical radio Telescope (FAST), as depicted in Fig. 1.4. The robot has a diameter of 600 m. Its end effector hangs from 6 steel cables suspended by six towers with a height of more than 100 m.

The use of a cable as the actuator improves successfully the workspace of the conventional parallel manipulator. Better dynamics and energy efficiency compared to the conventional parallel manipulator due to the lightweight cable are another benefit of the cable robot. However, due to unilateral properties, the cable provides only tension to the end effector, while the the conventional manipulator also applies pressure.



a. Winch-based system



b. Rail-based system [Bruckmann et al., 2011]

Figure 1.3: Common mechanisms for driving an end effector.

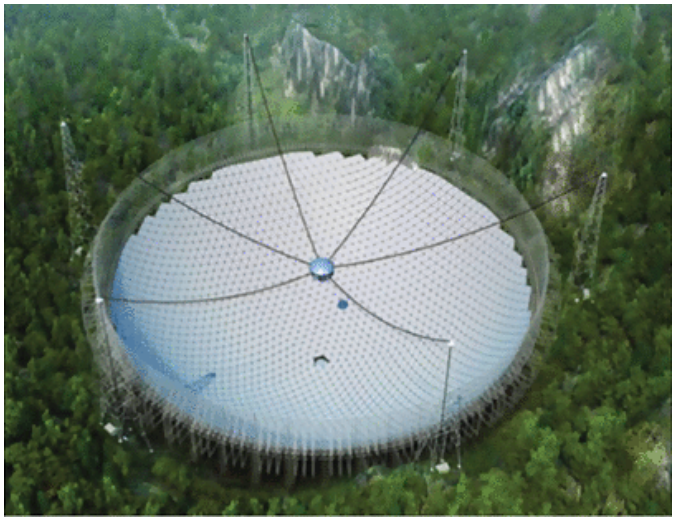


Figure 1.4: FAST, the biggest cable robot [Li et al., 2013].

Since the cable is only able to provide tension, the configuration of the cable robot is very important to achieve the desired number of degree of freedom (dof) and stability of the end effector. A fundamental classification of cable robots based on redundancy is proposed by [Ming and Higuchi, 1994] as follows:

1. Completely Restrained Positioning Mechanisms (CRPMs). This mechanism is kinematically redundant and requires at least $m = n + 1$ cables, where m is the number of cables and n is the number of degrees of freedom. The prototype SEGESTA[1] as depicted in Fig. 1.5 is an example of a CRPM.

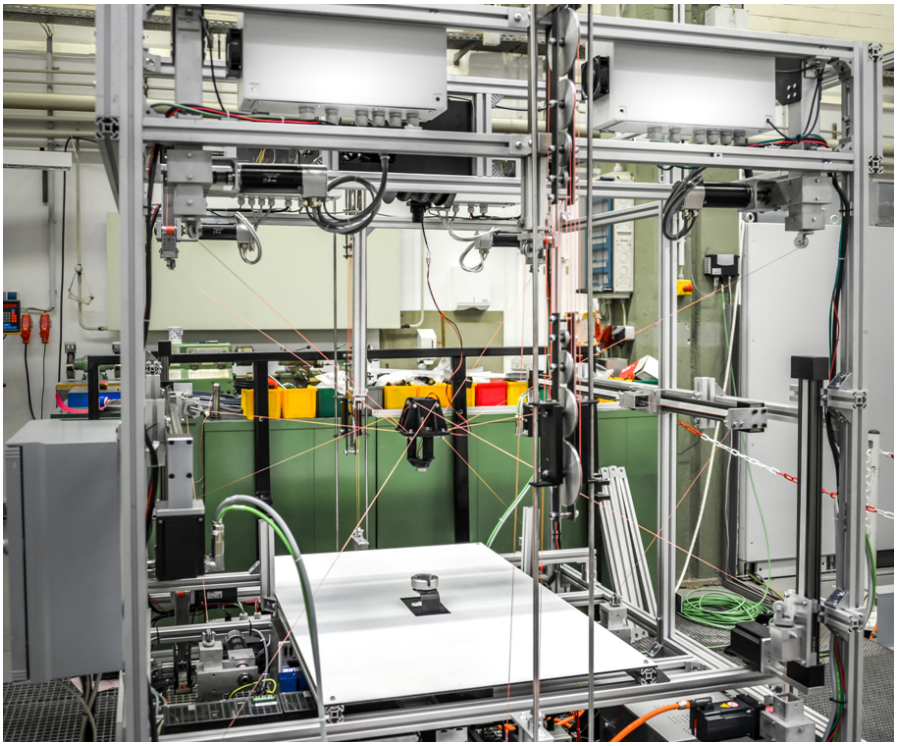


Figure 1.5: SEGESTA Version 2 [Reichert et al., 2015a]

2. Incompletely Restrained Positioning Mechanisms (IRPMs). This mechanism needs extra force (e.g. gravity) on the platform in order to put all cables under tension to stabilize the end effector. An example of this mechanism is the cable robot CABLEV[2] (Fig. 1.6).

Then, the category of CRPMs is extended by Verhoeven [2004] as follows

1. Completely Restrained Positioning Mechanisms (CRPMs), where $m = n+1$, and

2. Redundantly Restrained Positioning Mechanisms (RRPMs), if $m > n + 1$.

[1] Abbreviation of the German title, *Seilgetriebene Stewart-Plattformen in Theorie und Anwendung*, or in English, "Cable-based Stewart Platforms in Theory and Applications".
[2] Abbreviation of CABle LEVitation

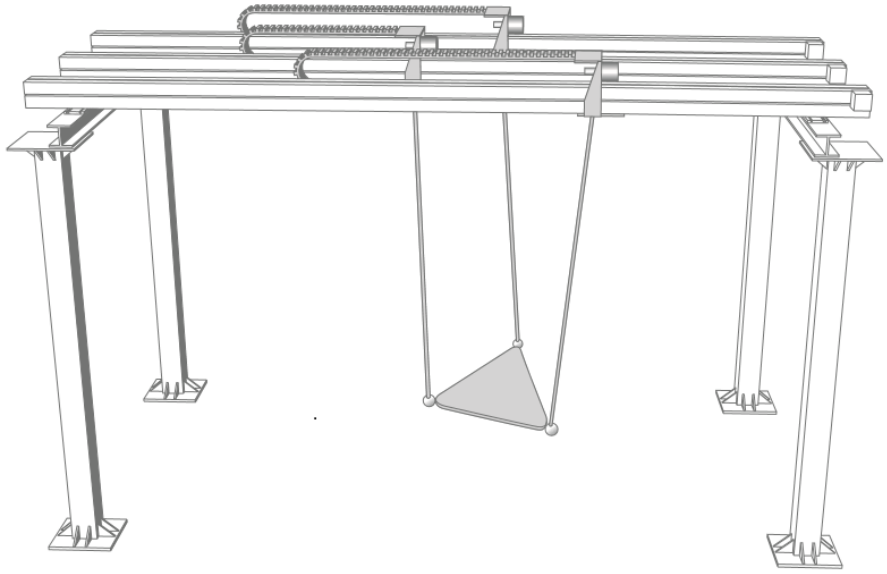


Figure 1.6: CABLEV [Bruckmann et al., 2008].

A further cable robot classification based on controlled dof is proposed by Verhoeven [2004] in order to avoid duality, because CRPMs/RRPMs can be transformed to IRPMs if the end effector is located at a specific position and orientation (pose), where external wrenches are required to obtain a positive force (tension) for all cables [Bruckmann et al., 2008]. The classification is given in the following:

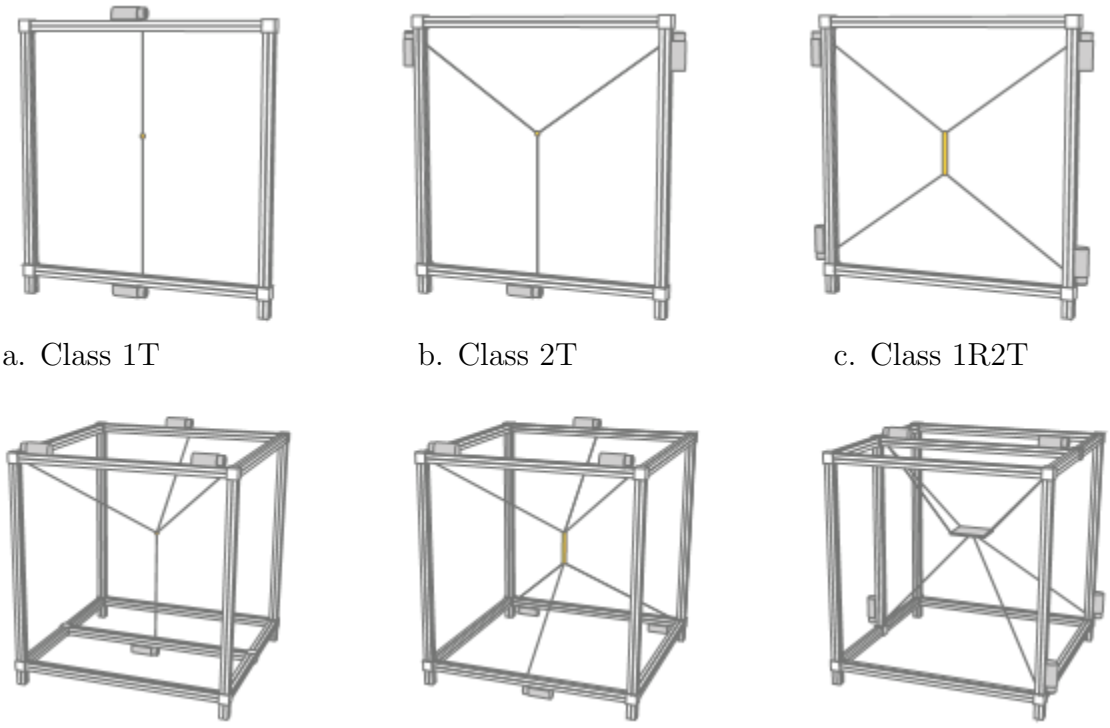
1. 1T: linear motion of a point,
2. 2T: planar motion of a point,
3. 1R2T: planar motion of a body,
4. 3T: spatial motion of a point,
5. 2R3T: spatial motion of a beam,
6. 3R3T: spatial motion of a body,

where T stands for translational and R stands for rotational. The classification of CRPMs is illustrated in Fig. 1.7. Verhoeven [2004] claims that this classification covers all possible dof of the end effector.

In the Chair of Mechatronics of the University of Duisburg-Essen, research about cable robots is being carried out under the following research projects:

1. SEGESTA, supported by DFG[3] under research project number HI 370/18. The fundamental research was developed within this research including the

[3]Abbreviation of the German title the *Deutsche Forschungsgemeinschaft*, or translated in English, the 'German Research Foundation'.



a. Class 1T b. Class 2T c. Class 1R2T

d. Class 3T e. Class 2R3T f. Class 3R3T

Figure 1.7: Cable robot class [Bruckmann et al., 2008].

development of the early version of a cable robot prototype with a similar name to the research project title.

2. ARTIST[4], supported by the DFG under project number HI370/24-1 and SCHR1176/1-2. In this research, an easy-to-use synthesis algorithm for cable driven parallel manipulators was developed, which determines the optimum parameters for the geometry of the working area defined by the application. In addition, a precise and low-wear cable guide is proposed to reduce the friction between the cable and the structure where the cable leaves.

3. *Stewart–Gough-Plattform* within the framework *EffizientClusterLogistikRuhr*, supported by the BMBF[5]. Within this research framework, the cable robot is applied as an automated storage retrieval system in a warehouse. The main output of this research is a medium-size prototype called CABLAR.

4. CableBOT, supported by the European Union's 'Seventh Framework Programme' (FP7/2007-2013) under grant agreement n° 285404. The main pur-

[4] *Arbeitsraumsynthese seilgetriebener Parallelkinematikstrukturen*, or in English, 'Workspace Synthesis of Cable Based Parallel Kinematic Structure'.

[5] Abbreviation of the German title of the *Bundesministerium fuer Bildung und Forschung*, or translated into English as the 'Federal Ministry of Education and Research of the Federal Republic of Germany'

pose of this research is to develop a reconfigurable and modular cable robot for application in the maintenance and transport of large scale products.

In this work, the author focuses on the CABLAR prototype. The goal of this work is to determine the platform pose of CABLAR using additional sensors.

1.1 Cable robot for logistic application

Today, high racks and Automated Storage/Retrieval Systems (AS/RS) are widely used to store and handle industrial goods. In logistics, it is well known that this part of the chain influences the production costs. An example of the currently used AS/RS is depicted in Fig. 1.8. Concerning the use of energy and handling time, the

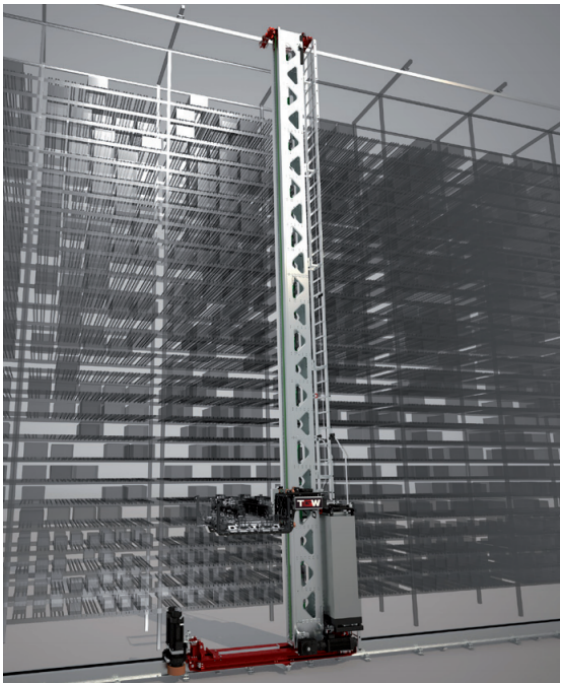


Figure 1.8: Conventional automated storage/retrieval system [Mustang, 2016].

weight of the moving parts itself must be decreased to improve the performance of the AS/RS, e.g. a low mass ratio between the cargo and the moving mechanisms. A low payload ratio leads to limited energy efficiency and motion capabilities as well as relatively high cycle times. Faster motions, lightweight and energy-saving solutions are desired to drastically reduce cycle times for the transport of the goods to satisfy ongoing climate change debates.

In 2010, the German Federal Ministry of Education and Research started a European logistics research initiative: the *EffizienzCluster LogistikRuhr* framework [Salah et al., 2011]. Within this framework, a sub-project called 'Storage

Retrieval Machine based on the Stewart–Gough-Platform' (SRM) was proposed with the aim of developing an alternative approach for AS/RS with efficient energy consumption while increasing the platform speed and acceleration using the advances of a parallel wire-based Stewart–Gough platform. A prototype called CABLAR, as shown in Fig. 1.9, has been built. According to Bruckmann et al.

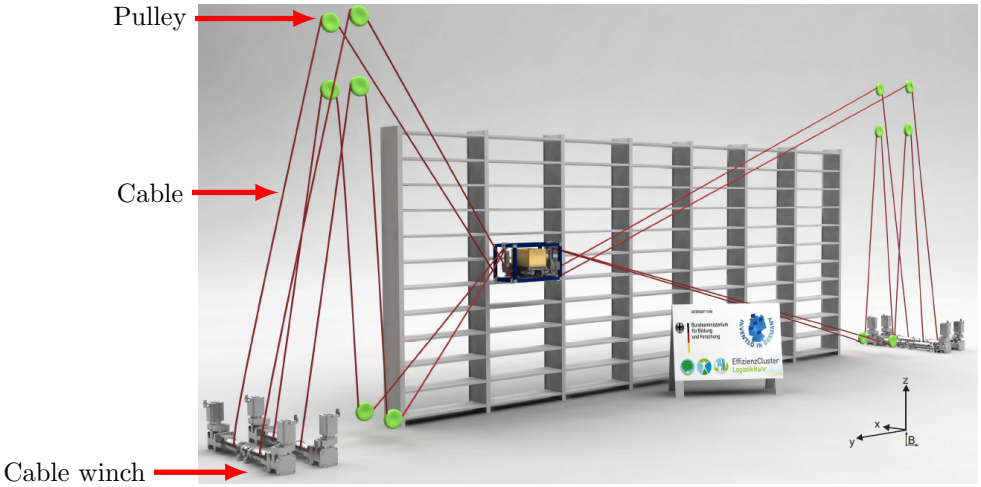


Figure 1.9: Rack feeder system based on the Stewart–Gough platform [Bruckmann et al., 2013].

[2013], an AS/RS based on a cable robot promises a better performance than one with a low mass ratio between the cargo and moving mechanisms. For instance, the moving mechanisms of conventional AS/RS for transporting 20 kg to 50 kg of cargo have a total weight of one to two tons, while the proposed system has only 150 kg. In addition, the cycle time (travel time) of the good storage process from the Input/Output (I/O) point to the rack as well as the retrieval process is improved significantly [Salah., 2013]. In order to give an overview for the reader, the importance of the system of CABLAR is described briefly in the following paragraph.

The CABLAR controller was developed under the Matlab/Simulink® environment, then compiled, to generate a TwinCAT 3 runtime module by a licensed TE1400 / TC3 Target for MATLAB/Simulink® compiler from Beckhoff Automation. A PC based automation software package called TwinCAT 3 was chosen as the real-time control system, which runs on an Industrial Personal Computer (IPC). The controller exchanges the data with other components, such as EtherCAT terminals, sensors, and actuators, under an Ethernet-based fieldbus system, namely EtherCAT®[6]. The components must have EtherCAT I/O to enable integration with the EtherCAT fieldbus system, otherwise an EtherCAT terminal is required. An example of a connection between sensors with an EtherCAT terminal is depicted in Fig. 1.10.

[6]Abbreviation of Ethernet for Control Automation Technology

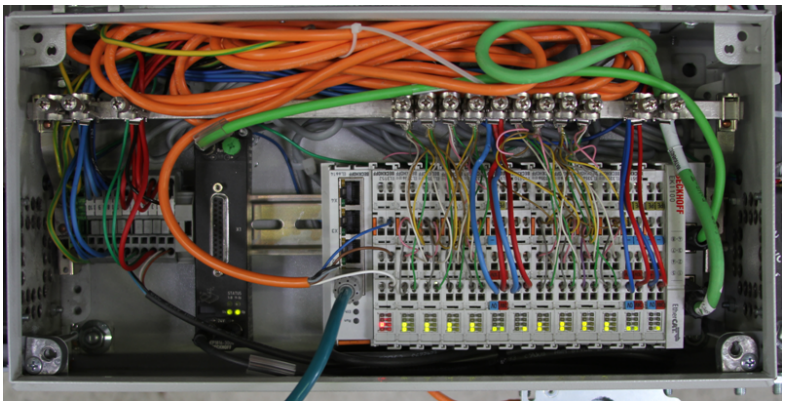


Figure 1.10: Connection of the sensors with EtherCAT terminals.

The plastic cable, type Dyneema®, is fixed at the force sensor type Megatron KM1101 K 5KN 0000 Z at one side, as depicted in Fig. 1.11. At the other side,

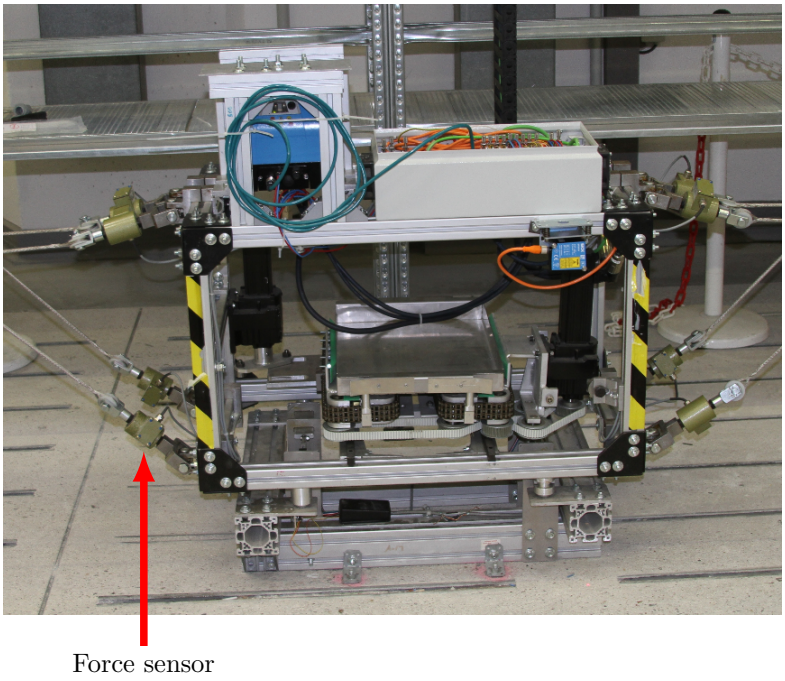


Figure 1.11: Platform of the CABLAR.

the cable is wounded at the winch through the pulley as shown in Fig. 1.9 and Fig. 1.12. The cable forces are measured by 8 force sensors installed between the platform[7] and the cables as depicted in Fig. 1.3.a and Fig. 1.11.

Eight SEW EURODRIVE motors of type CMP100M/BP/KY/AK1H/SBB with integrated rotary encoder were chosen as the actuator. This type of motor is able

[7] here the term platform is used instead of end effector

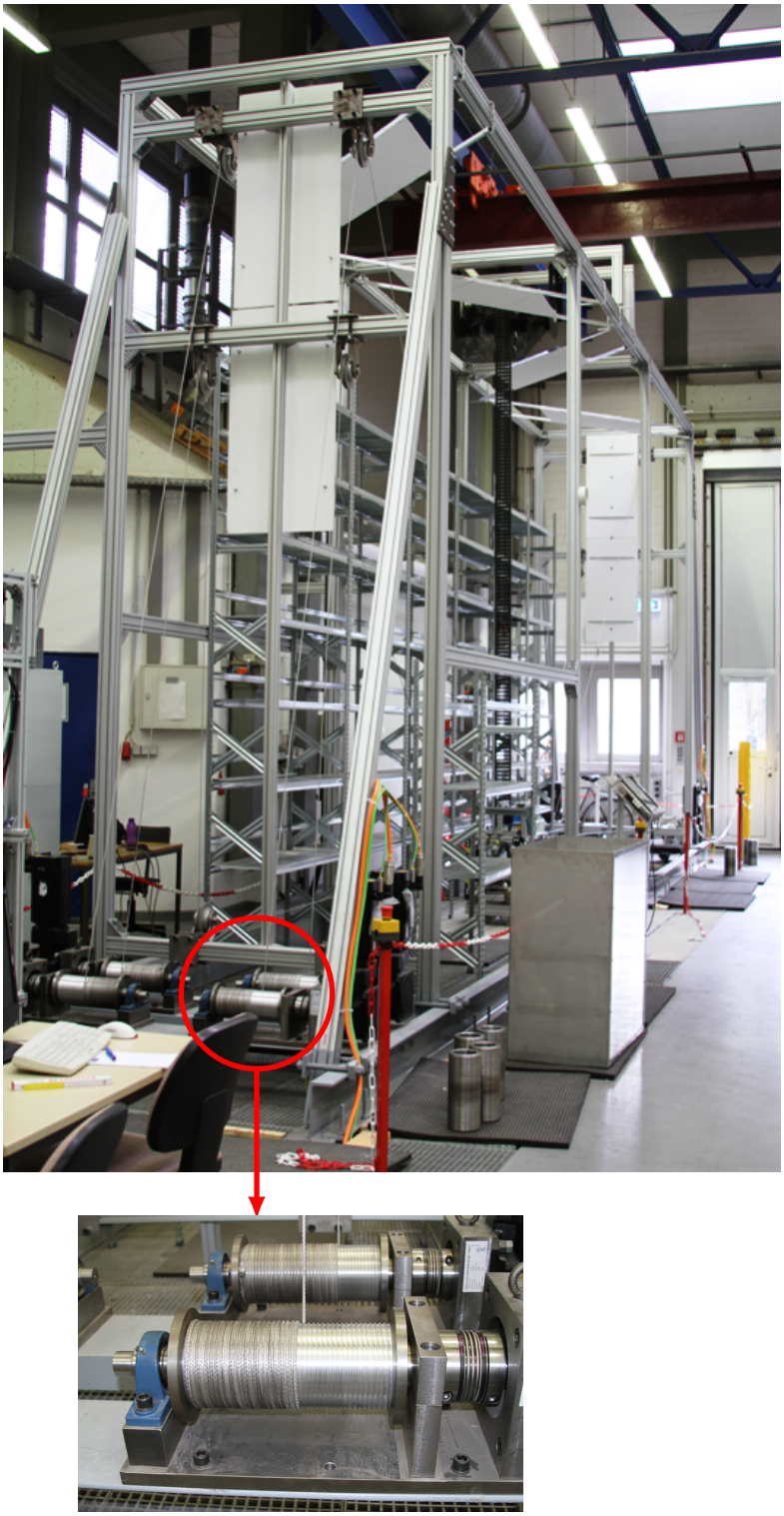


Figure 1.12: CABLAR prototype.

to generate up to 31 Nm torque and 14.6 kW electrical power. In addition, an intelligent servo controller with EtherCAT I/O supports a command mode with three available control options for the motor: torque control, position control, and velocity control. A gearbox transmits the power from the motors to the winch. Since the motors and the cable winches are fixed at the frame base (floor), concave roller pulleys are required to change the cable's direction and to guide the cable. In order to reduce the friction between the cable with the roller, the pulleys are pivoted at the arms. The arms are then fixed at the robot frame with a specific position, which depends on the desired robot workspace. The workspace calculation of CABLAR including cable and pulley configuration can be found in more detail in [Lalo, 2013].

The operational procedure of CABLAR is depicted in Fig. 1.13. The first step is

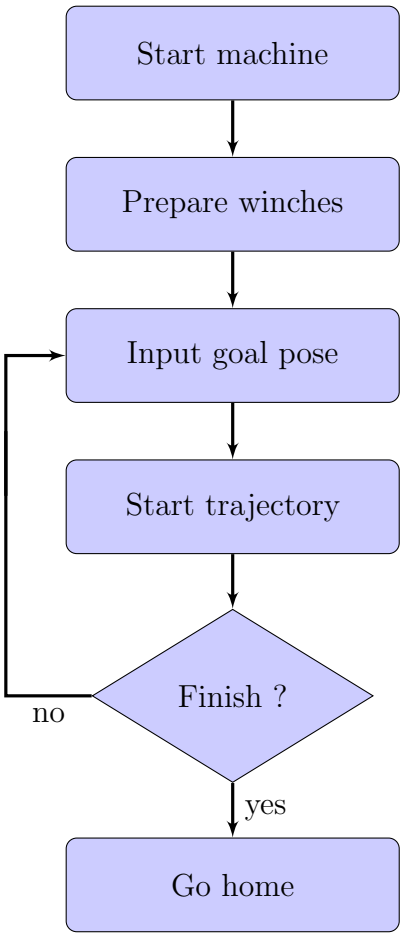
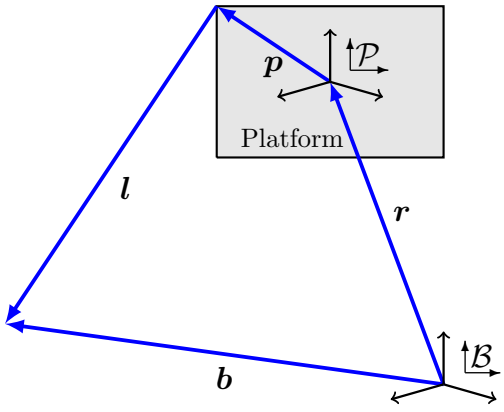


Figure 1.13: CABLAR operational flowchart.

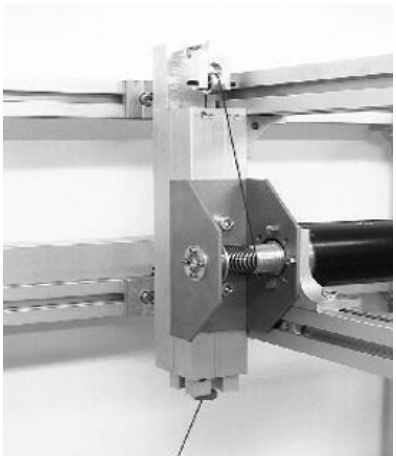
to prepare the machine. In this step, the preparation procedures for all hardware, such as sensor check, sensor calibration, etc., are executed. The preparation of the winches follows as the second step, with the purpose of calibrating the lengths of the winches and cables. Within this step, the platform is locked at home position

by a mechanism, while a defined force is applied to all cables to ensure all cables are under the proper tension in order to avoid sagging. In the same step, the encoder position of the motors are recorded when all the cables are under tension, and is called the encoder reference position. In addition, the lengths of the cables are determined by inverse kinematics and called the cable offset. Since the platform is locked, the cable can be assumed to be a perfectly rigid body. After the winches are prepared, the platform is unlocked automatically and the goal pose is given on the Graphical User Interface (GUI). The platform's motion is started by pushing the start trajectory button on the GUI. The trajectory planner generates the desired trajectory and the controller ensures the platform takes the desired platform trajectory. After the goal pose is achieved, a new goal pose can be input on the GUI or the platform returns to the home position if no further motion is required.

It must be highlighted here that the inverse kinematics requires a kinematics model, which is shown in Fig. 1.14.a. The contact point of the cable with its guidance



a. Kinematic of cable robot



b. Winch design of the SEGESTA prototype [Verhoeven, 2004]

Figure 1.14: Prototype and kinematic modeling of cable robot.

is denoted by $\boldsymbol{l}$, which is constant if point-shape-hollow guidance is used. The small ceramic eyes are an example of point-shape-hollow guidance, as depicted in Fig. 1.14.b. The platform's position and orientation referenced to the base coordinate system $\mathcal{B}$ are represented by $\boldsymbol{r}$ and ${}^{B}\boldsymbol{R}_{P}$, respectively. The connection point of the cable with the platform with respect to the platform coordinate system $\mathcal{P}$ is denoted by $\boldsymbol{p}$. Since all other vectors are known, the vector of the actuator/cable $\boldsymbol{l}$ can be computed easily. It should be noted here that the use of point-shape guidance in a cable robot is not reliable, due to the high friction between the cable and the guidance. For that reason, the pulley is chosen as the guidance in CABLAR. Although the complexity of the kinematic modeling becomes higher, it is

still solvable. More details about the CABLAR kinematics modeling can be found in [Bruckmann, 2010, Bruckmann et al., 2008, Lalo, 2013].

The inverse kinematics is necessary not only for preparing the winches but also for the controller. CABLAR and the new Segesta prototypes have the same control concept, namely augmentedPD-Control (APC), depicted in Fig. 1.15. This concept

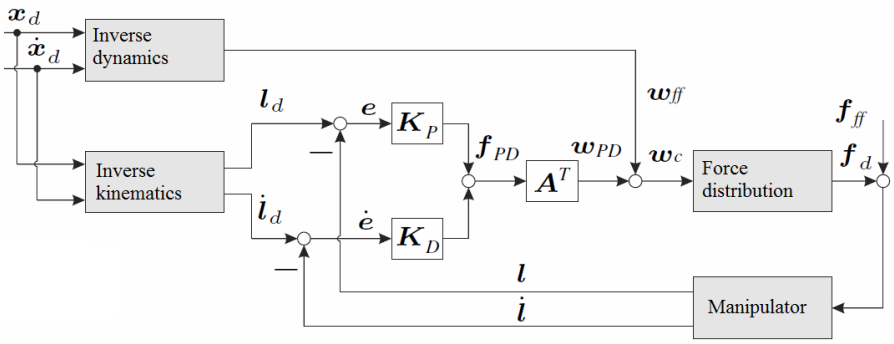


Figure 1.15: Block diagram of augmented PD-Controller [Reichert et al., 2015b].

is designed based on joint coordinate and PD-Controller[8] [Reichert et al., 2015b]. This controller maintains the cable's lengths to keep the platform motion close to the desired/generated trajectory. To have a better overview, the control concept of CABLAR will be explained in the following paragraph.

The platform's motion must be started from a known pose, namely home position. Soon after the start trajectory button is pushed, the trajectory planner generates the desired trajectory $\boldsymbol{x}_d$ and velocity $\dot{\boldsymbol{x}}_d$ based on the goal pose and desired motion time. Both outputs from the trajectory planner are fed into the inverse kinematics and inverse dynamics block. The vector $\boldsymbol{w}_{ff}$ of forces and torques, called the wrench, is intended to compensate the platform dynamics including the disturbances, and is computed by inverse dynamics. Meanwhile, the inverse kinematics calculates the desired cable's lengths $\boldsymbol{l}_d$ and velocities $\dot{\boldsymbol{l}}_d$. Then, the errors of the cables $\boldsymbol{e}$ and $\dot{\boldsymbol{e}}$ are calculated by comparing $\boldsymbol{l}_d$ and $\dot{\boldsymbol{l}}_d$ with the actual cable lengths $\boldsymbol{l}$ and actual cable velocities $\dot{\boldsymbol{l}}$. $\boldsymbol{l}$ and $\dot{\boldsymbol{l}}$ are obtained by evaluating the encoder reference position and cable offset at the beginning of the controller's running, and evaluating the actual cable length, prior cable length, actual encoder position, and prior encoder position for the next. The knowledge of all the parameters of the robot allows the determination of $\boldsymbol{l}$ and $\dot{\boldsymbol{l}}$. The cable elongation is taken into account. The cable is modeled as a string, and its stiffness is determined by evaluating its Young's modulus, length, and diameter.

For the cable robot application, the gain matrix is assumed as the cable stiffness. According to the assumption, the multiplication of the cable length difference ($\boldsymbol{e}$ and $\dot{\boldsymbol{e}}$) with the gain matrix $\boldsymbol{K}_P$ and $\boldsymbol{K}_D$ yields the vector of forces $\boldsymbol{f}_{PD}$, which

[8] Abbreviation of Proportional and Differential, a well known control loop

is later multiplied with the so called structure matrix $\boldsymbol{A}^T$, resulting in the wrench $\boldsymbol{w}_{PD}$. The sum of $\boldsymbol{w}_{PD}$ and $\boldsymbol{w}_{ff}$ is fed to the force distribution algorithm to compute the optimum force $\boldsymbol{f}_d$. To compensate the friction forces and the moment of inertia of the winch drum, the defined force $\boldsymbol{f}_{ff}$ is added. The electrical motor realizes the desired force via torque.

1.2 Problem statement

Merlet [2013] addresses several open issues in cable robot research:

- kinematics, e.q. the distance between the cable anchor point at the base and the platform is determined from the cable length with the assumption that the cable under the proposer tension. In fact, that assumption is not always valid,
- singularities, e.g. the influence of the cable configuration on the cable force. The singular cable configuration leads to the infinite cable force,
- workspace & planning,
- redundancy & control,
- dynamics.

Those issues can be the reason for imperfections in CABLAR's performance in the operation, e.g.

1. by visual observation during CABLAR operation, the actual platform pose seems not at the desired pose[9].
2. the platform trajectory has a parabolic shape on the horizontal motion where a straight trajectory is defined,
3. the platform does not return exactly to the home position after a few tens of minutes of robot operation,
4. the platform pose error[10] is directly proportional to the robot's operation time.

Presumably, the error is accumulated from several sources, such as imperfections in the assumptions, floating point error, measurement error, changes in material properties due to time in operation and environmental aspects, etc. Currently, the accumulated error is eliminated for safety reasons (e.g. to avoid a crash between the platform and the rack, or the platform's rolling over) after a certain period of operation by bringing the platform to the home pose and performing

[9]The platform poses are observed in static condition
[10]The error is defined as the deviation from the desired pose

step *prepare winches*[11]. For the purpose of time efficiency, this procedure should be performed only when necessary, which can be realized if the measured pose is available. Moreover, the measurement data is also important to benchmark the controller and to calibrate the robot. For that reason, the platform pose somehow must be measured.

There are two existing methods to determine the platform pose:

1. indirect measurement using proprioceptive sensors, e.g. forward kinematics [Fang, 2005, Merlet, 2004, Merlet and Alexandre-dit Sandretto, 2015, Pott and Schmidt, 2015], which can be stated briefly: the platform pose is determined by evaluating all the cable's lengths,

2. direct measurement using external sensors, such as a camera system [Babaghasabha et al., 2014, Chellal et al., 2015, Dallej et al., 2012] or laser tracker [dit Sandretto et al., 2013, Kraus et al., 2012].

Forward kinematics has attracted more attention from cable robot researchers due to its high frequency and the fact that no additional sensor is necessary. However, the following factors influence its performance:

1. All cables are assumed under tension. Unfortunately this assumption is not valid for the entirety of the time of the robot's operation, due to a reason in the force distribution algorithm [Merlet, 2013]. As a result, the actual wire's lengths do not represent the distance from the cable anchor point on the platform to the contact point on the cable guidance, which influences the accuracy of the forward kinematics.

2. Cable sag. In many works, the cable mass and elasticity are neglected in the kinematics model. This assumption is valid only for lightweight cables and small scale robots. The influence of this problem is similar to that of the point above.

3. Manufacturing aspects. Due to tolerances in machining and assembly, the parameters in design are not similar to the prototype. This problem can be solved by calibration, either internal or external [dit Sandretto et al., 2013]. However, a calibration with proprioceptive sensors requires forward kinematics. On the other hand, external sensors such as a laser tracker or vision-based system are costly. For instance, a Bonita system from Vicon with 6 IR cameras are desired for a cubic robot with a side of 3 m [Chellal et al., 2015]. Meanwhile, the laser tracker can not measure all the desired kinematic parameters due to the desired point's being hidden behind another part.

[11] *prepare winches* is the second step in the CABLAR operational flowchart as depicted in Fig. 1.13

The implementation of a laser tracker or camera system promises a better measurement result compared to forward kinematics, but is not worth being implemented for CABLAR because this cable robot is intended to be a commercialized product. The sensor will impact the selling price of the product significantly. An alternative measurement method is proposed by the author to replace the existing method. In the present work, a 2 Dimensional (2D) laser scanner combined with a particular design of a reflector and Inertial Measurement Unit (IMU) is proposed to determine the platform pose. Both sensors cost less than 20% of a laser tracker or camera system.

The main objectives of this thesis are modeling, simulation, and experiment, which is elaborated in the following points:

1. *To select the suitable sensors and to interface the sensors with the robot system.* A suitable sensor has been selected for this particular application in CABLAR by considering several aspects, such as measurement frequency, robustness, interface, and price. Two system architectures for data transfer between the laser scanner and the controller system have been tested and compared to select the best one. Since the laser scanner is delivered without a software development kit, a communication program and measurement data conversion program have to be prepared. Moreover, the internal time in the laser scanner must be synchronized with the host computer.

2. *To design the reflector pattern.* A reflector with special pattern is necessary in order to generate unique measurement data that can be used to extract all translational components of the platform position vector and two of the three rotational components.

3. *To develop the kinematic model.* The kinematic model of the cable robot system, including a laser scanner, has been developed. The model was later used to simulate the platform during operation, either while stationary or during motion, considering the effect of the velocity on the measurement data. The simulation result is an imitation of the measurement data from the laser scanner. An algorithm to compensate the influence of the platform motion on the simulation result/measurement data imitation has been developed.

4. *To extract the straight line parameters and to determine the platform pose.* The simulation result from the step above is processed to obtain the straight line parameters which correspond with the reflectors. Several straight line extraction methods are combined to achieve the best result, which is computationally efficient. The kinematics model, based on a mathematical model, and the special pattern of the reflector, were developed to determine all translational components of the platform's position and two components the of platform's orientation. Since the laser scanner requires a certain time for scanning the object and data transfer, as well as for data processing, an algorithm for delay compensation is important. The result is the actual platform pose.

5. *Experiment and validation.* The proposed method and sensors, including sensor architectures, were tested on the cable robot prototype. Then, the determined platform position was validated with direct measurement of the platform position.

The platform pose determined in this work is not currently intended for the purpose of robot control. The determined pose is important for the following purposes:

1. Evaluating the platform positioning by the robot controller to the desired pose.
2. Providing the platform pose to a robot calibration process.
3. As the reference value for error elimination by bringing the platform to the home pose.

1.3 Structure of this thesis

The present thesis consists of six chapters and is arranged as follows: the automation software TwinCAT 3, the laser scanner, and the IMU are explained briefly in Chapter 2. Two laser scanners are compared and the suitable laser scanner for the application in this work is selected based on several criteria. Two system architectures for the data transfer, with the purpose of exchanging the data between the sensors with the software TwinCAT 3, are explained and compared. Equally important is the procedure to synchronize the internal time of the host computer with the laser scanner.

In Chapter 3, the kinematic model of the cable robot, including the laser scanner, is explained. Since the simulation of the measurement data during the platform motion is also desired, the platform velocity is considered in the mathematical modeling. Then, a mathematical model to compensate for the effect of velocity on the measurement data is described. The limitation of the laser scanner for this particular application is discussed. A reflector with a special pattern is proposed with the aim that the measurement data be unique and can be used to determine all components of the platform position vector. A mathematical model is developed to extract the x-component of the platform position from the measurement data.

Chapter 4 explains several popular methods to extract the straight line parameters from a set of data. The desired data which corresponds with the reflector are extracted, then segmented for further processing. A strategy to obtain the best straight line parameters by combining several straight line extraction methods is described. Then, the platform pose is determined based on the straight line parameters. The determined platform position must be compensated to obtain the actual platform pose in automation software.

Chapter 5 presents the simulational and experimental results. The comparison of the simulation results with and without the velocity effect are presented. Three different platform velocities, with and without the velocity compensation algorithm, are simulated, with the aim of finding the characteristics of the algorithm. In the experiments, several scenarios are executed. The measurement data from two different system architectures of data transfer are collected and compared in order to choose the best one. Then the proposed method to determine the platform pose is compared with another measurement method for validation purpose. After validation, the measurement result from the proposed method is collected during the platform's motion. The time delay compensation algorithm is verified.

The result of this work is summarized in Chapter 6. The scientific and technical contribution is described. Recommendations for future work are also given in that chapter.

CHAPTER 2

Software, Hardware, and Their Interface

The robot controller is operating on the automation software Beckhoff TwinCAT® 3 real time controller system. The automation software communicates with the hardware such as sensor, actuator via Ethernet field bus system EtherCAT®. Meanwhile, the proposed sensors in this study do not have EtherCAT® interface. A converter is required to transfer the measurement data from the sensor interface into EtherCAT system. The hardware, software and the system architecture to transfer the measurement data are explained and discussed in the following sections.

2.1 Automation Software TwinCAT 3

TwinCAT® 3 is a real-time PC based automation software package from Beckhoff Automation. This program is integrated as an extension in the Microsoft Visual Studio® environment; its GUI is illustrated in Fig. 2.1. An application program, called a module, such as the controller module or a measurement module, can be created and generated by the TwinCAT PLC, NC, C/C++ programming language and Matlab/Simulink®. The nodes of the programming environment, except for Matlab/Simulink®, are depicted under the Solution Explorer part in Fig. 2.1. The generated application/module is uploaded onto the TwinCAT 3 software in the Sub Node "TcCOM Objects" located under the node "SYSTEM" of Solution Explorer. "TcCOM Objects" is shown in Fig. 2.2.

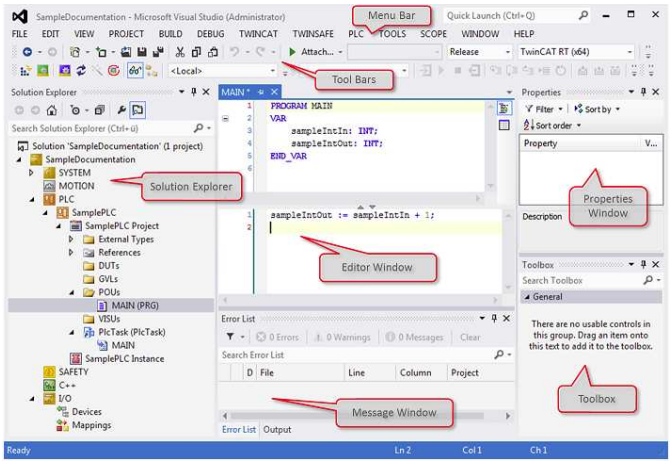


Figure 2.1: Main window of the TwinCAT 3 engineering environment [Beckhoff, 2015c].

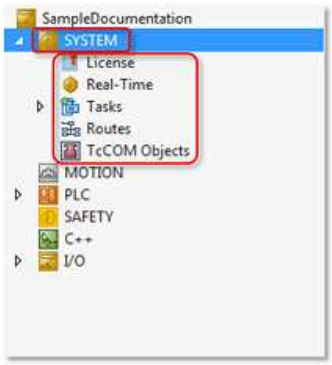


Figure 2.2: Node tree of TwinCAT 3 [Beckhoff, 2015c].

The I/O devices (fieldbus devices) are added to TwinCAT 3 under node I/O, shown in Fig. 2.2. A GUI to operate the application program is prepared either by TwinCAT PLC or by another program outside of TwinCAT 3, for instance C++. The so-called TwinCAT Automation Device Specification (ADS) allows data exchange between another software and TwinCAT 3. To allow the objects (modules, I/O devices, sensors, actuators, etc.) to exchange data with each other, a connection is required, by linking the input/output variable of the object. An example of task I/O variables is shown in Fig. 2.3.

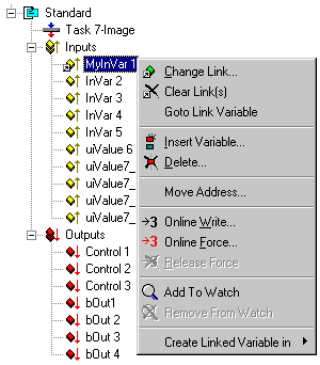


Figure 2.3: Variable link of an object [Beckhoff, 2015c].

The base time[1] and CPU limit[2], is specified under the subnode Real-Time. The fastest available base time is 50 μs and if not necessary should not be set below 1 ms [Beckhoff, 2015b]. The cycle time of the module can be a multiple of the base time. If necessary, different base times can be applied for different CPUs.

TwinCAT 3 provides extensive support of multi-core[3] systems. In the Base CPU management under the subnode Real-Time, illustrated in Fig. 2.4, the limit of each CPU should be adjusted properly to achieve the optimum performance of the TwinCAT software and PC. There are several options for the CPU limit available in the dialog box. In the same subnode, the CPU number for handling each task (if more that one task exists) is chosen. A balanced distribution of the CPU load will benefit the PC performance.

The configuration of the controller system must be done under **Config Mode**. If the configuration is completed, the system is ready to run in real-time under **Run Mode**. To finish the run mode, one can choose either **Stop Mode** or return to Config Mode.

[1]Base Time is the time frame for a constant interrupt, where the scheduler pauses Windows® and provides time for TwinCAT Real-Time tasks Beckhoff [2015b]

[2]This value defines the maximum percentage that the Base Time scheduler can use for Real-Time tasks. The remaining time is available for Windows [Beckhoff, 2015b]

[3]In TwinCAT, “cores” are considered as “CPUs”

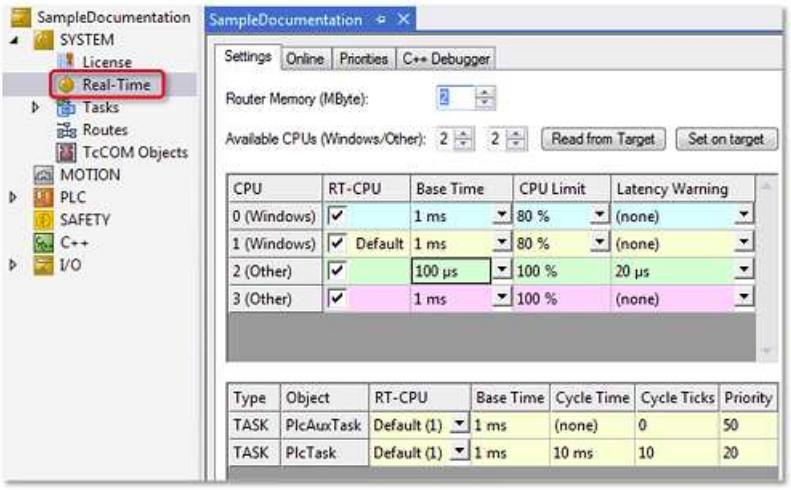


Figure 2.4: TwinCAT SYSTEM Sub Node "Real-Time" [Beckhoff, 2015b].

2.2 2D Laser Scanner

2.2.1 Selection

Laser is an acronym for **L**ight **A**mplification by the **S**timulated **E**mission of **R**adiation. Since its invention in 1960, the laser has been used for scientific and commercial applications [Kneubühl and Sigrist, 2005]. One of the commercial uses of the laser is the 2D laser scanner, which is widely used nowadays in various applications, such as:

- industrial application for in- and out-door building safety and security [Sick, 2015a], in ports application (e.g. anti-collision systems), traffic application (e.g.: vehicle classification, speed monitoring, etc.), area monitoring in museums, galleries, and other buildings [Sick, 2015c],
- agricultural applications in measuring plants, growth rate, tree volume, tree count, 3D imaging, and pattern recognition [Lee and Ehsani, 2008],
- and in robotics research for navigation, mapping and obstacle detection with the aim of collision avoidance [Cho and Hong, 2010, Hoeller et al., 2014, Jiménez and Naranjo, 2011, Liu et al., 2013, Qiu and Han, 2009, Saito et al., 2013, Wender et al., 2005].

Two suitable laser scanners for use in this work with almost the same specifications are available on the market: Sick LMS500 and Hokuyo UTM-30LX. Their specifications are given in Table 2.1. The Sick LMS500 has a faster data transfer rate (using Ethernet interface), the ability to adjust its angular resolution, and a faster cycle time or higher measurement frequency. These advantages and the wide use of the Sick product in industry and research were the reasons for choosing the Sick LMS500 in this research.

Table 2.1: Technical data of SICK LMS500and Hokuyo UTM-30LX [Sick, 2015b] and [Hokuyo, 2012]

Parameter	Symbol	Sick LMS500	Hokuyo UTM-30LX
Aperture angle	Θ_{max}	190 °	270 °
Angle resolution	$\Delta\Theta$	0.167 °, 0.25 °, 0.333 °, 0.5 °, 0.667 °, 1 °	0.25 °
Number of measurement results/beams	n_b	$\frac{\Theta_{max}}{\Delta\Theta}$	1080
Operating range	t_{max}	80 m	60 m
Systematic Error	e_s	±25 mm (1 m... 10 m) , ±35 mm (10 m... 20 m)	±30 mm 0.1 m... 10 m
Cycle time (Depends on $\Delta\Theta$)	τ_c	max 10 ms	25 ms
Scanning time	τ_s	$\frac{\tau_c}{2}$	unknown
Transfer time	τ_t	$\frac{\tau_c}{2}$	unknown
Measurement result	–	Polar or Cartesian coordinate systems	Polar coordinate systems
Interface	–	USB (500 kBaud), Ethernet (10/100 MBit/s) and Serial (500 kBaud)	USB (12 MBit/s)

Lee and Ehsani [2008] studied the characteristics of the Sick LMS200 and Hokuyo URG-04LX. Although the sensor types are not similar, their result is worth considering because the Sick LMS200 is the predecessor of the Sick LMS500. One of the results of their study for this research is the sensor warm-up settling time. The laser scanner Sick LMS200 must be warmed up for at least 53 minutes before the measurement is started.

2.2.2 Operating principle of the laser scanner

The operation of a 2D laser scanner is based on the time-of-flight principle, as illustrated in Figs 2.5 and 2.6. A pulse of light (a laser beam) is emitted by a

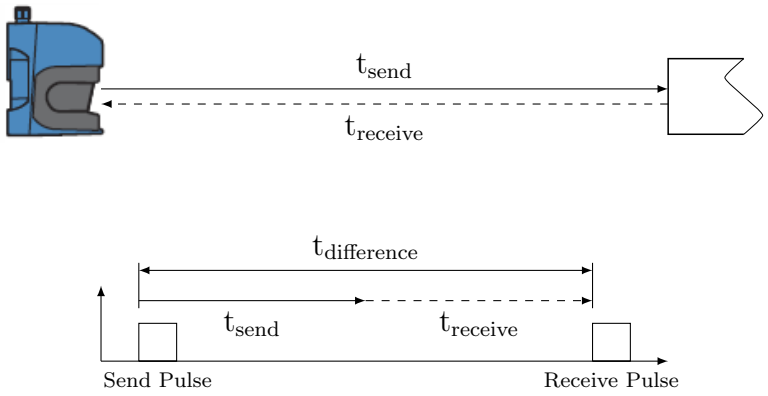


Figure 2.5: Principle of operation for time of flight measurement [Sick, 2015b].

laser diode over a certain period of time. If a reflector (an object) exists on the laser beam's path, the pulse will be reflected to the sender and captured by the receiver. The time counter records the time when the pulse of light leaves the sender (outgoing) and the time when the reflected pulse of light is captured by the receiver (incoming). The difference in time is proportional to the distance between the sensor and the object. This sensor has an internal rotating mirror to deflect the laser beam for measuring the surrounding distance radially, as depicted in Fig. 2.6. The rotating mirror is driven by an electric motor with an angular encoder. The motor speed will be adjusted according to the measurement frequency. The angular encoder has the function of triggering the measurements at the adjusted angle resolution.

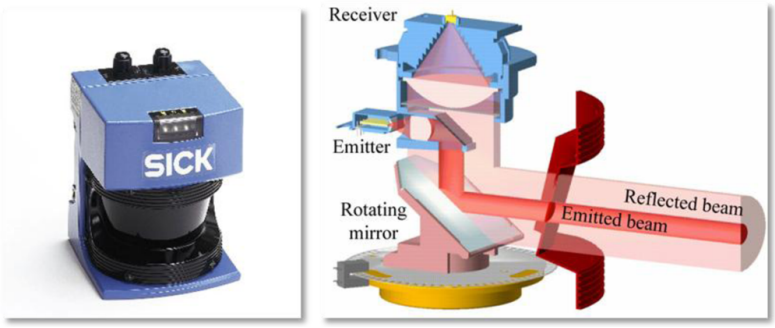


Figure 2.6: Sick LMS200 and its working principle [Sanz-Cortiella et al., 2011].

2.3 Interfacing the Sick LMS500 with TwinCAT 3

The Sick LMS500 has three interfaces: USB, serial, and Ethernet. The USB interface is used only for the purpose of setup and is not recommended for real time application. To select a suitable interface for the application in this study, the serial and Ethernet were to be tested and compared.

2.3.1 Serial interface

The serial interface is one of the popular interfaces for real time applications. The EtheCAT terminal serial interfaces allow communication of the laser scanner with the automation software in real time. Several protocol converters from various companies are available on the market, but there is no product which has a matching baud rate (data transfer rate) with the laser scanner. The product with the closest baud rate with the laser scanner is Unigate CL-EtherCAT (UCE, Fig. 2.7), manufactured by the company Deutschmann Automation. The UCE consists of

Figure 2.7: UCE Protocol Converter [Deutschman, 2013].

the following hardware [Deutschman, 2014]:

- Electrically isolated EtherCAT®-Interface,

- EtherCAT® controller ET1100,
- Microprocessor 89C51RD2,
- RAM and EEROM,
- Serial interface (RS232, RS485 and RS422) to connect the device with a sensor,
- Debug interface (RS232) to connect the device with the PC.

After communication with their sales department, the company Deutschmann was able to assemble the product with a special baud rate.

The UCE protocol converter is controlled by a script [Deutschman, 2014], which was written under the free software "Protocol Developer" from the Deutschmann Company [Deutschman, 2013]. The main window of the protocol developer is depicted in Fig. 2.8. The script consists of commands, classified as

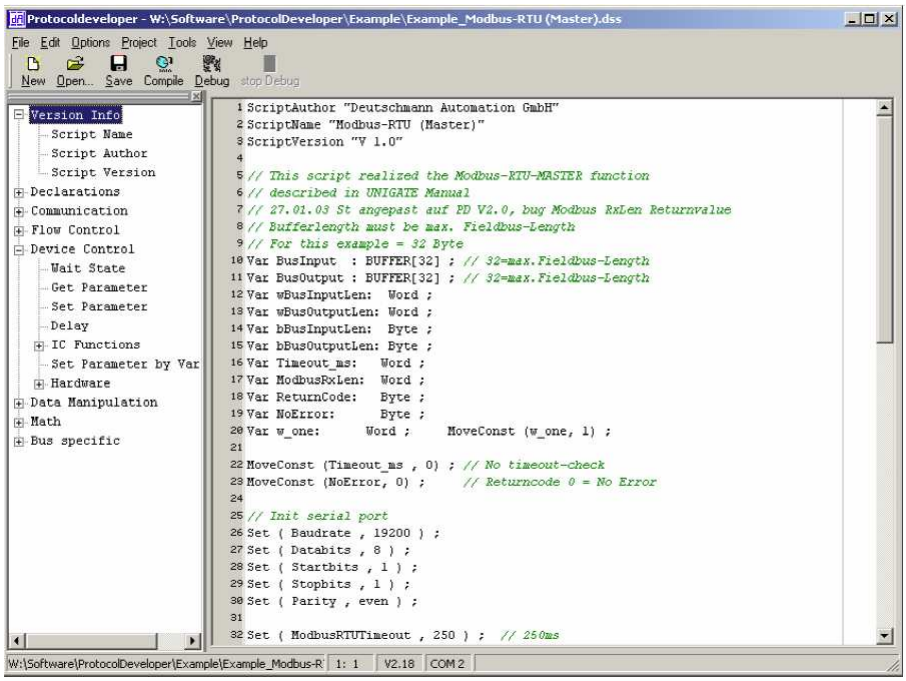


Figure 2.8: Main window of the protocol developer [Deutschman, 2013].

follows.

- **Declarations**: variable declaration,
- **Flow control**: sub-function calls, jumps, branches,
- **Math**: mathematical functions, data conversions,
- **Communication**: send and receive data,
- **Bus specific**: the command are placed that enter bus-specific values,

- **Version info**: text, issued by the Gateway in its switch-on message,
- **Data manipulation**.

The template of the script is provided by the manufacturer, but can be modified. This software allows the user to program the device according to the application.

The baud rate of the serial port depends on the processor's crystal frequency. The following equation determines the baud rate of the protocol converter according to the crystal frequency.

$$\begin{aligned} BaudIst &= \frac{F32}{K} \\ F32 &= \frac{CrystalFrequency(Hz)}{32} \\ K &= \text{round}\frac{F32}{BaudSoll} \end{aligned} \tag{2.1}$$

Here, $BaudIst$, $Baudsoll$, 32, and round are the baud rate realized by the UCE, the desired baud rate, a constant number, and the commercial round off, respectively. The difference between the desired and realized baud rates (baud rate error e_B) is given, in percentage, by

$$e_B = \left(abs \left(\frac{BaudIst - BaudSoll}{BaudSoll} \right) \right) 100. \tag{2.2}$$

The standard Unigate CL-Ethernet uses 40 Hz, therefore the baud rate is adjustable to 2.4, 4.8, 9.6, 19.2, 57.6, 312.5, and 625 kBaud [Deutschman, 2014].

For instance, the standard UCE is adjusted to work at 38400 baud. According to Eq. 2.1 and Eq. 2.2, the UCE will realize a data transfer rate of 37879 baud and a baud rate error of 1.3573 %. According to [Deutschman, 2014], an error below 2 % can be accepted in practice.

To match the data transfer rates of the UCE and the laser scanner, the UCE's manufacturer replaced the standard crystal oscillator (40 MHz) with one of 16 MHz for the purpose of getting the baud rate error below 2 %. This substitution decreased the UCE's processor speed by 60 % and required adapting the firmware.

To investigate the performance of the UCE in terms of its ability to transfer the data from the laser scanner to the TwinCAT software, a simple experiment with a setup as in Fig. 2.9 was performed. The LMS500 is connected to the protocol converter through an RS422 cable. The Industrial Ethernet/Patch cable exchanges the data between the protocol converter and an industrial PC with the installed automation software TwinCAT 3. For debugging purposes, and to control the laser scanner, the debug interface on the UCE is connected to the serial port on a PC with protocol developer software inside.

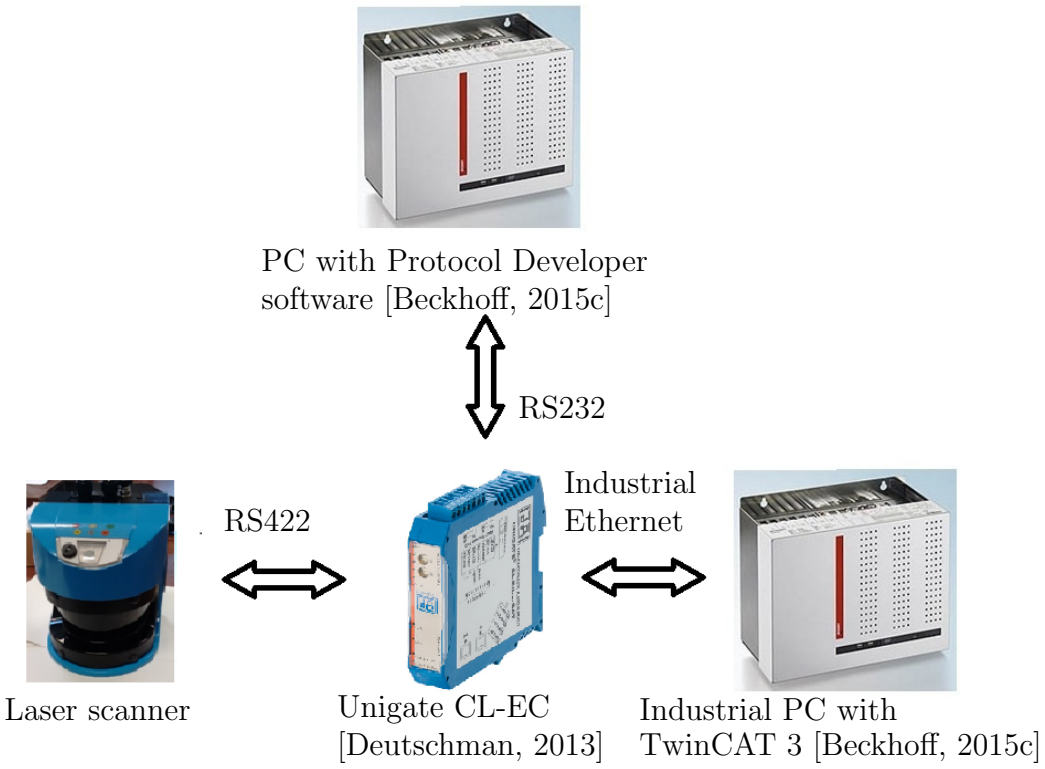


Figure 2.9: System architecture of protocol converter for testing.

In this experiment, the laser scanner was controlled by the script in protocol converter software and adjusted as follows:

- 75 Hz measurement frequency (13.32 ms for each cycle time),
- continuous measurement mode,
- 180 ° aperture angle, and
- 1 ° angle resolution.

There were 372 bytes of data received by the protocol converter for each measurement/scan from the laser scanner. Due to the limitations of the EtherCAT XML device description in the TwinCAT software, the UCE is only able to transfer a maximum of 255 of the 372 bytes of data in a cycle. This means the data transfer for each measurement/scan is completed in two cycles.

The result of this simple experiment above, which can only be observed visually on the protocol developer software, led to the conclusion that the protocol developer was not able to transmit all the measurement data from the laser scanner maximally (only circa 10 Hz among 75 Hz frequency). This conclusion was strengthened by the statement in [Deutschman, 2014] that the script processing time is approximately 0.5 ms for each line and the processor itself has several tasks, such as the following:

- sending and receiving data at the Debug-interface,
- sending and receiving data at the serial interface,
- sending ad receiving data at the Filedbus interface,
- task controlled via internal clock (1 ms)(e.g. flashing of a Light Emitting Diode), and
- processing the script.

Moreover, the laser scanner manufacturer stated, in personal communication with the author, that the Programmable Logic Controller (PLC) software such as TwinCAT 3 was not able to handle the measurement data from the Laser Scanner because of the huge amount of data. For instance, a 25 Hz measurement frequency with 0.25 ° angular resolution and 180 ° aperture angle resulting 18000 $^{\text{bytes}}/_{\text{second}}$.

To avoid the transfer of such a huge amount of data from the UCE to the controller, an additional peripheral with the following requirements,

- light and small size (placed on the end effector),
- robust against shocks,
- real-time capable, and
- able to handle the measurement data from laser scanner,

is required to be placed between the laser scanner and the UCE with the aim of extracting the desired data from the measurement data.

One of the suitable peripherals is the embedded system PC104 speedMOPSlcdPM from the company Kontron, shown in Fig. 2.10. This PC/104 has a 1.8 GHz

Figure 2.10: PC/104 speedMOPSlcdPM [Kontron, 2005].

Pentium M [Kontron, 2005] processor and 256 MB RAM, operated by the Microsoft Disk Operating System (MS-DOS). Several I/O, such as 2x USB2.0, 2x RS 232 port, 1x 10/100 BaseT Ethernet, 1x PC/104 I/O expansion modules, are available. The laser scanner requires the RS 422 port, which is not available on this PC/104

module. Nevertheless, it is possible to add an I/O expansion module to the PC/104. The system architecture with the additional peripheral is depicted in Fig. 2.11. One

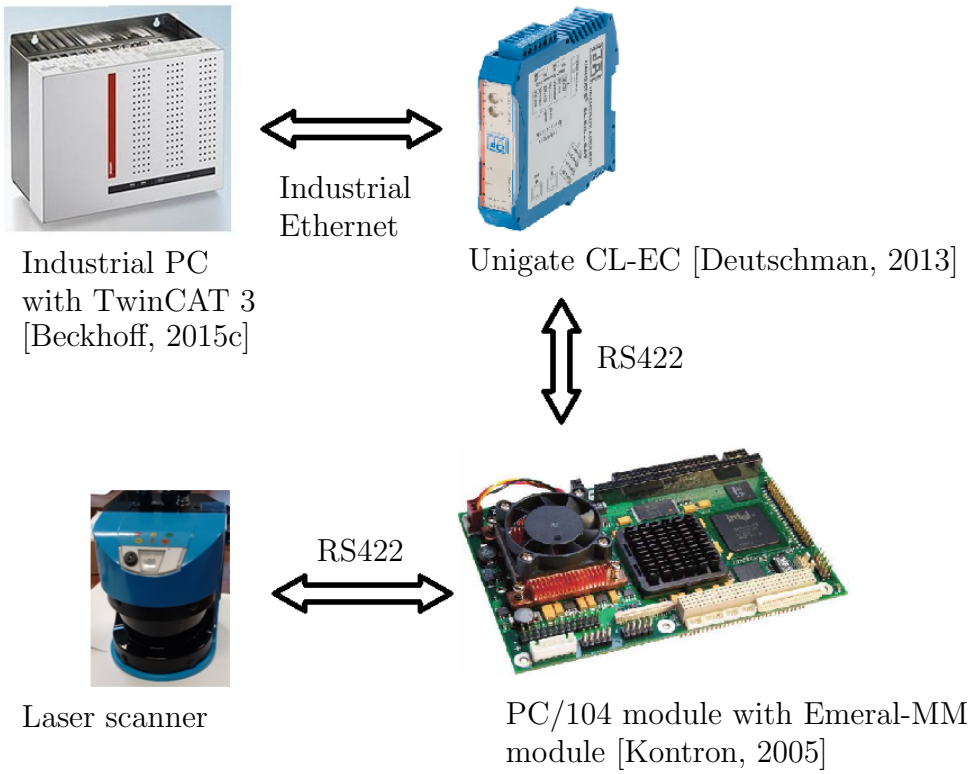


Figure 2.11: System architecture of the data transfer using serial interface.

of the suitable RS422 I/O expansion modules for the PC/104 is the Emerald-MM module with 4x tuneable RS232/RS422/RS485 ports, as shown in Fig. 2.12. in two versions, normal and fast baud rate. Both versions have no matching baud rate with the laser scanner, but the manufacturer can fulfill a customer's request by replacing the crystal oscillator. According to the manufacturer, the 8 MHz crystal oscillator generates an exact 500 kbaud rate.

To control the physical system (laser scanner) and to communicate with the UCE, a Simulink project with Emerald MM I/O block and data processing blocks was prepared. The placement of the PC/104 in the system decreases the amount of data to be transmitted to the UCE significantly. Only 60 bytes instead of 372 bytes are transmitted to the protocol converter for each scan.

2.3.2 Ethernet interface

The Ethernet interface on the LMS500 offers a high data transmission rate in comparison with the other available interfaces. For data transmission from the laser scanner through the EtherCAT system to the host computer, an Ethernet

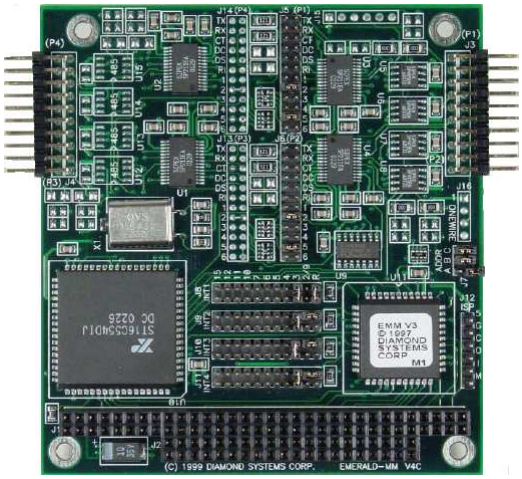


Figure 2.12: Serial ports Emerald-MM [Emerald-MM, 2003].

Figure 2.13: Ethernet switch port terminal EL6614 [Beckhoff, 2015a].

interface is required. Fig. 2.13 shows the Ethernet switch port terminal EL 6614 from the Beckhoff company.

In EtherCAT technology, the Ethernet terminal has the function of decentralizing the connection of the Ethernet terminal of the host PC to the EtherCAT terminal network. In other words, this terminal is only an extension of the normal PC Ethernet. The terminal does not bring the data to the real-time traffic data in the TwinCAT system. The manufacturer is able to design a system that transmits two types of Ethernet data simultaneously (normal Ethernet data and TwinCAT Ethernet data). According to the manufacturer [Beckhoff, 2015a], the EtherCAT relays the Ethernet communication of the connected devices fully transparently and collision free.

The LMS500 manufacturer does not provide a software development kit but provides a ready to use program, namely SOPAS Engineering Tool (SOPAS ET), without the possibility of being modified. This software allows the user to configure aspects of the laser scanner such as: the aperture angle, resolution angle, measurement frequency, etc., easily on a user friendly GUI. The measurement data is displayed in this software and can be collected if necessary. Unfortunately, there is no possibility of requesting the real-time measurement data from this software

for further data processing in a real time environment, which is required in this research. Although the determined pose is not intended for control purposes in this thesis, real-time measurement is important for the following purposes.

1. A huge amount of measurement data from the laser scanner must be processed in real time in order to reduce the amount of data which is captured for the robot calibration.

2. As the reference value for bringing the platform to the home pose, the determined pose must be available in real time.

To fulfill the requirements of the further data processing, *interface program* serves the following required tasks:

- To send the command to the laser scanner,
- to receive the measurement data in ASCII and convert into decimal numbers,
- to compute the translational and rotational components of the end effector pose $\boldsymbol{r}$ from the measurement data, and
- to transmit the data to the TwinCAT system.

In point of view of the data exchange between this interface program and TwinCAT 3, this solution is enabled because TwinCAT 3 has the so-called TwinCAT Automation Device Specification (ADS) with the aim of exchanging the TwinCAT data with other software, such as C++. This solution also promises the benefit of overcoming the drawback stated by the laser scanner manufacturer as mentioned in subsection 2.3.1, by carrying out some of the calculations inside a C++ program to decrease the amount of data to be transmitted to the TwinCAT 3 system.

Furthermore, the Ethernet interface allows a client (such as a PC) to communicate with the LMS500 over Transmission Control Protocol/Internet Protocol (TCP/IP). A promising solution to meet the requirements above is Windows Socket (WinSock)/ Socket programming under C++. According to Microsoft® [4], WinSock allows the programmer to create internet applications to transmit application data across a wire, independently of the network protocol being used.

The architecture of the sensor concept is illustrated in Fig. 2.14. The communication between WinSock with Sick LMS500 is established across EtherCAT system via a single patch cable, which is also used by all the sensors and actuators for communication with the controller (automation software). The ADS and Winsock code in the programming language C++ is given in Appendix D.

The laser scanner has two measurement modes: continuous and non-continuous. The telegram for measurement in continuous mode is sent only once to operate the laser scanner continuously until a telegram to stop the measurement is received.

[4] https://msdn.microsoft.com/en-us/library/windows/desktop/ms740673(v=vs.85).aspx

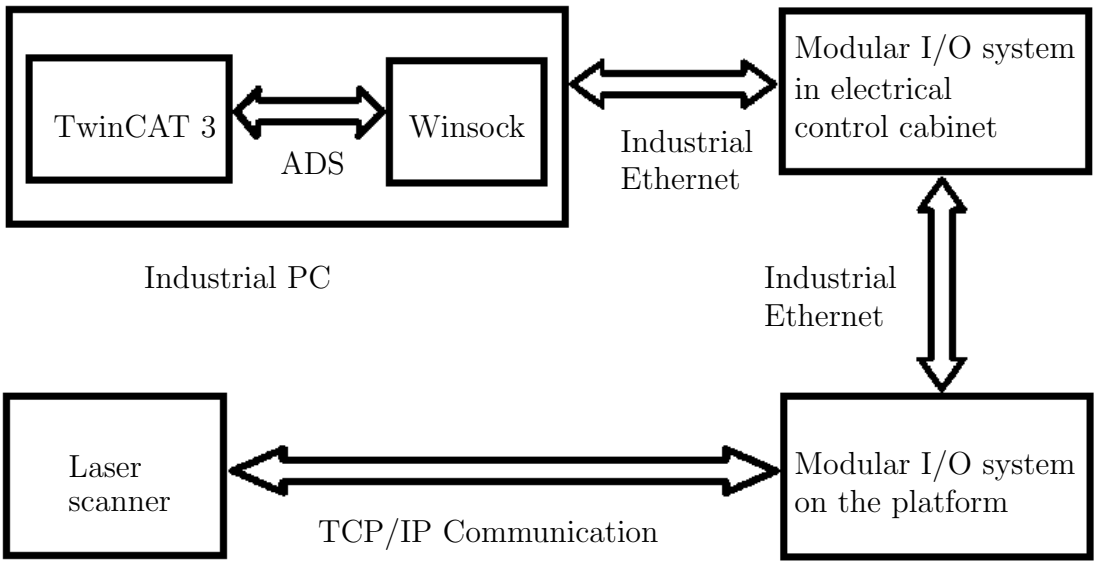


Figure 2.14: System architecture of the data transfer using Ethernet interface.

In the other mode, the laser scanner conducts the measurement only once in non-continuous mode.

To choose the suitable mode for this work, a simple experiment has been conducted. The experiment was carried out with 2 kHz controller frequency (500 μs base time). 80 % of first CPU and 50 % of second CPU were limited to the TwinCAT 3 software. More details about the CPU limitation is mentioned in Section 2.1 and shown in Fig. 2.4. The laser scanner was adjusted with an angle resolution of 0.25 ° and measurement frequency of 35 Hz. The operating system Windows and the proposed program to drive the laser scanner, including receiving the measurement data, were running under the Windows API on the second CPU in a similar PC to that where the controller was running.

A window was prepared to display the size of the incoming packet data (measurement data) in the interface program. One can see the incoming packet data on the prepared window. According to the displayed incoming packet data, the continuous mode was not chosen for this work. In continuous mode, the measurement data is not always for a single scan. The measurement data is received by the interface program often in multiple packets[5]. Since the incoming packet data is not always from a single scan, the data conversion for the further step becomes more complex and requires more time. The low allocated CPU capacity for the interface program and jitter[6] in an IP network are the possible reasons for this situation.

[5]One packet is defined as the measurement data from one scan.
[6]Jitter is defined as a variation in the delay of the received packets [Cisco, 2016].

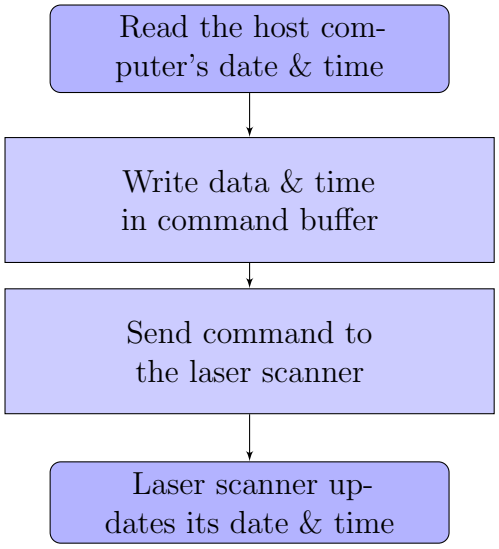


Figure 2.15: Time synchronization process.

2.4 Time synchronization and measurement delay

The laser scanner outputs the measurement result together with the timestamp in a packet of data. The timestamp is the time when the measurement was started, with the data format of year, month, day, hour, minute, second and microsecond. Since the laser scanner has no real time clock inside, the internal time has to be set when the laser scanner is started or restarted. To set the internal time, a telegram with a special format has to be sent to the laser scanner.

The measurement process inside the laser scanner until the measurement data is available, as well as the data processing by the proposed method, requires a certain amount of time. In addition, the data processing runs in the `C++` environment under the Microsoft Windows Application Programming Interface (Windows API) where its frequency depends on the actual resources of the computer (current memory load, processor load, etc.). Consequently, a delay for the reasons already mentioned causes the measurement result to no longer be the current platform position for such a condition where the measurement performed during the platform motion. A solution is proposed in this study to estimate the current position of the platform but the measurement delay must be known.

To obtain the measurement delay, the time in the laser scanner must be synchronized with the system time of the computer so the timestamp of measurement and the current system time in computer are compared to determine the measurement delay. The flowchart of the time synchronization process is depicted in Fig. 2.15. The time synchronization proceeds in the `C++` environment under the Microsoft Windows Application Programming (Windows API) Interface. The time synchronization process itself requires time processing until the time in the laser scanner is

updated. Consequently, the current time in the host computer is not the same as the current system time of the laser scanner. The time difference between the host computer and the laser scanner will be determined later from the experimental result in Section 5.2

2.5 Inertial Measurement Unit

The Inertial Measurement Unit (IMU), a Microstrain 3DM-GX1®, shown in Fig. 2.16, consists of a 3-axis accelerometer, 3-axis gyroscope, and 3-axis mag-

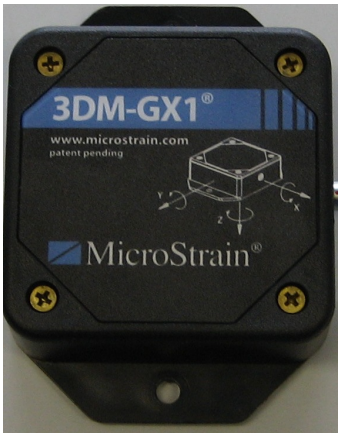


Figure 2.16: Microstrain 3DM-GX1®.

netometer. The sensor has a size of 65x90x25 mm and weight of 75grams. This IMU communicates with the host through an RS232 or RS485 serial port with a transfer rate of 19.2 kBaud, 38.4 kBaud and 115.2 kBaud. The port and transfer rate are software selectable. The measurement result is requested from the sensor by a specific command, as given in Table 2.2. There are two possible measurement results that can be obtained from the 3DM-GX1®: Instantaneous and Gyro-Stabilized. Instantaneous means that the measurement result is transmitted without any filter meanwhile Gyro-Stabilized includes the complementary filter to cancel the noise in the signal processing.

The 3DM-GX1® has an on-board processor to handle the following tasks [Microstrain, 2015a]:

1. Convert raw sensor outputs into digital form.
2. Scale sensor outputs into physical units (including temperature, alignment, and G-sensitivity compensation). This provides the instantaneous vector quantities.
3. Compute the Gyro-Stabilized vector quantities using the complementary filtering algorithm.

Table 2.2: Commands data quantities of 3DM-GX1®. Source: Microstrain [2015b]

Data Quantities	Command in Hexadecimal
Send Raw Sensor Bits	0x01
Send Gyro-Stabilized Vectors	0x02
Send Instantaneous Vector	0x03
Send Instantaneous Quaternion	0x04
Send Gyro-Stabilized Quaternion	0x05
Send Instantaneous Orientation Matrix	0x0A
Send Gyro-Stabilized Orientation Matrix	0x0B
Send Gyro-Stabilized Quaternion & Vectors	0x0C
Send Instantaneous Euler Angles	0x0D
Send Gyro-Stabilized Euler Angles	0x0E
Send Gyro-Stabilized Quaternion & Instantaneous Quaternion	0x12
Send Gyro-Stabilized Euler Angles & Acceleration & Rate Vector	0x31

4. If the host has issued a command byte (or if operating in continuous mode), compute the appropriate response data and transmit (e.g., Euler Angles, Matrix, Quaternions, Vectors, etc.).

The processor's timer tick is adjustable by modifying a value located in memory number 238, 240, 242, and 240 of EEPROM[7] of the sensor. For instance, the EEPROM for this research is specified as

$$\begin{aligned} \text{EEPROM } 238 &= 4 \\ \text{EEPROM } 240 &= 10 \\ \text{EEPROM } 242 &= 250 \\ \text{EEPROM } 246 &= 2. \end{aligned}$$

and the timer tick is calculated as $4 \cdot 10 \cdot 250 \cdot 2 \cdot 2 \cdot 10^{-7}\text{s} = 2\,\text{ms}$. Then, the measurement frequency is computed as follows.

$$\frac{1000\,\text{ms}}{2\,\text{ms}} = 100\,\text{Hz}. \tag{2.3}$$

The measurement frequency can be adjusted to another frequency, depending on the quantity of data requested from the laser scanner. The adjustment procedure can be found in more detail in Microstrain [2015b].

[7] **E**lectrically **E**rasable **P**rogrammable **R**ead-**O**nly **M**emory

2.6 Concluding remarks

The automation software for robot controller and its communication system were explained. Two system architectures to transfer the measurement data from the laser scanner to the robot communication system are proposed and will be tested in the experiment on the real prototype. The The laser scanner type Sick LMS500 and the IMU type Microstrain 3DM-GX1® were chosen as the measurement tools.

CHAPTER 3

Modeling of CABLAR

3.1 Kinematics Modeling of the CABLAR

In this section, the kinematic model as the basis for imitating the laser scanner measurement data within the robot workspace during operation is developed. The mathematical model to imitate the platform position while stationary and in motion, including its algorithm to compensate the measurement data, is developed. Then, the mathematical model to compute the intersection between the laser beam and the reflector is also developed.

3.1.1 End effector position vector

The kinematic of CABLAR is depicted in Fig. 3.1, while the frame dimensions are in Table 3.1. The platform pose with respect to the inertial system $\mathcal{B}$ is denoted

Table 3.1: The dimension of the SRM frame

Parameter	Symbol	Value
Width	Δx	1.78 m
Length	Δy	10.47 m
Height	Δz	5 m

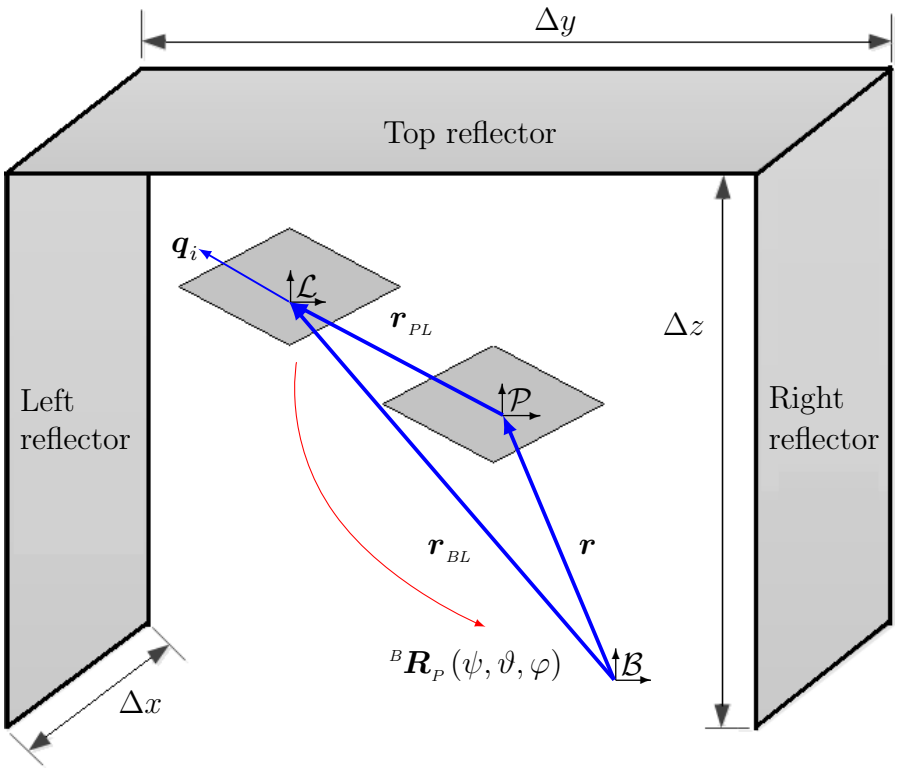


Figure 3.1: Kinematics of CABLAR.

by

$$\boldsymbol{x} = [\,{}^{B}\boldsymbol{r}^{T}\ \varphi\ \vartheta\ \psi]^{\mathrm{T}} \tag{3.1}$$

where

$${}^{B}\boldsymbol{r} = [x\ y\ z]^{\mathrm{T}} \tag{3.2}$$

is the position of the platform with respect to $\mathcal{B}$. The rotation matrix of the platform with respect to $\mathcal{B}$ is denoted by ${}^{B}\boldsymbol{R}_{P}$. According to Spong et al. [2006], the rotation matrix in spatial motion is given by

$${}^{B}\boldsymbol{R}_{P}(\psi, \vartheta, \varphi) = \boldsymbol{R}_{z}(\psi) \cdot \boldsymbol{R}_{y}(\vartheta) \cdot \boldsymbol{R}_{x}(\varphi), \tag{3.3}$$

with

$$\boldsymbol{R}_{z}(\psi) = \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{3.4}$$

$$\boldsymbol{R}_{y}(\vartheta) = \begin{bmatrix} \cos\vartheta & 0 & \sin\vartheta \\ 0 & 1 & 0 \\ -\sin\vartheta & 0 & \cos\vartheta \end{bmatrix}, \tag{3.5}$$

$$\boldsymbol{R}_x(\varphi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\varphi & -\sin\varphi \\ 0 & \sin\varphi & \cos\varphi \end{bmatrix}, \tag{3.6}$$

where ψ, ϑ and φ are the yaw angle about the z-axis, pitch angle about the y-axis, and roll angle about the x-axis, respectively. The rotation matrix of Eq. 3.3 is known as the yaw-pitch-roll.

The laser scanner is fixed on the platform with identical orientation. Their coordinate systems are denoted by $\mathcal{L}$ for the laser scanner and $\mathcal{P}$ for the platform. Since the orientation of these coordinate systems are identical, so that ${}^{B}\boldsymbol{R}_{L} = {}^{B}\boldsymbol{R}_{P}$. The position of the laser scanner with respect to $\mathcal{P}$ is denoted by ${}^{P}\boldsymbol{r}_{PL}$ while the position of the laser scanner with respect to $\mathcal{B}$ is given by

$${}^{B}\boldsymbol{r}_{BL} = {}^{B}\boldsymbol{r} + {}^{B}\boldsymbol{r}_{PL} \tag{3.7}$$

with

$${}^{B}\boldsymbol{r}_{PL} = {}^{B}\boldsymbol{R}_{L}\,{}^{L}\boldsymbol{r}_{PL}. \tag{3.8}$$

Eq. 3.7 is the basis for developing the mathematical model in order to simulate the laser scanner during operation. If the platform is in stationary position while the laser scanner completes a scan/measurement, then ${}^{B}\boldsymbol{r}_{BL}$ is identical for all laser beams in the single scan. However, the velocity effect must be taken into account for the simulation where the platform is moving. The following subsection will discuss this issue.

3.1.2 Velocity effect

Subsection 2.2.2 mentioned that the laser scanner has an internal rotating mirror to deflect the laser beam for the distance measurement. This means that the laser beams are deflected not at the same time, but sequentially. Consequently, the measurement data of a scan is not acquired at the same time. The sequential deflection of the laser beam in a scan influences the measurement results if the laser scanner and the objects are not stationary relatively to each other. As a result, the real origin of the laser beam for a scan is not the same. In this subsection, the mathematical model to compensate for the influence on the measurement of the object–laser scanner motion is developed.

To simplify the modeling in this subsection, the end effector velocity is assumed constant. Including the velocity effect in the mathematical model requires the knowledge of the laser scanner's working principle. The internal rotating mirror inside the laser scanner takes a certain time τ_s to complete the deflection of n_b laser beams for every single scan. Both parameters are known from the laser scanner

configuration in Table 2.1. Thus, the time difference from one laser beam to the next is given by

$$\tau_{inc} = \frac{\tau_s}{n_b}, \tag{3.9}$$

which is also called the time increment.

Suppose the end effector moves from a start position $\boldsymbol{r}_s$ to an end position $\boldsymbol{r}_e$ with a constant speed v_s. The distance between the positions is computed by

$$s = \|\boldsymbol{r}_e - \boldsymbol{r}_s\| \tag{3.10}$$

which takes time

$$\tau = \frac{s}{v_s} \tag{3.11}$$

to complete the motion from the start to the end position.

The number of scan measurements during the motion is calculated by

$$n_s = \frac{\tau}{\tau_c}. \tag{3.12}$$

Since the end effector speed is assumed constant, this yields

$$\boldsymbol{v}_e = \frac{\boldsymbol{r}_e - \boldsymbol{r}_s}{\tau} \tag{3.13}$$

where $\boldsymbol{v}_e$ is the end effector velocity.

$\boldsymbol{v}_e$ is required to compute the platform position for each laser beam in a scan. Furthermore, the platform position is necessary in the modeling of the intersection of the laser beam with the reflector. However, the direction of each laser beam with respect to the laser scanner coordinate system $\mathcal{L}$ must be known, which is described in the following subsection.

3.1.3 Laser beam vector

The laser scanner measures the distance to the object radially by rotating the internal mirror. Each laser beam in the scan has a direction vector specified by the adjusted angular resolution. The direction vector of each laser beam is modeled in the following paragraph.

Consider the index set of all indices of the laser beam $\mathcal{I}_M = \{0, 1, 2, \cdots, n_M\}$. The vector of the i-th laser beam $\boldsymbol{q}_i$ with respect to $\mathcal{L}$ is given by

$${}^{L}\boldsymbol{q}_i = \begin{bmatrix} 0 & \cos(\delta_i) & \sin(\delta_i) \end{bmatrix}^{\mathrm{T}} \quad , \quad i \in \mathcal{I}_M, \tag{3.14}$$

where

$$\delta_i = \Delta\Theta \cdot i. \tag{3.15}$$

$\Delta\Theta$ is the adjusted angular resolution of the laser scanner as given in Table 2.1.

${}^{L}\boldsymbol{q}_i$ is decomposed in $\mathcal{B}$ by

$${}^{B}\boldsymbol{q}_i = {}^{B}\boldsymbol{R}_{L}\,{}^{L}\boldsymbol{q}_i. \tag{3.16}$$

By now, the position of the laser scanner with respect to the robot inertial system $\mathcal{B}$ and the direction vector of all laser beams are known. In the next subsection, the intersection of the laser beam with the reflector is modeled.

3.1.4 Modeling the intersection of the laser beam and reflector

The intersection of the laser beam with the reflector will be discussed in this section. The mathematical model is based on idea of the intersection of a vector with a plane. Suppose the frame of CABLAR is a cuboid, as illustrated in Fig. 3.2. The

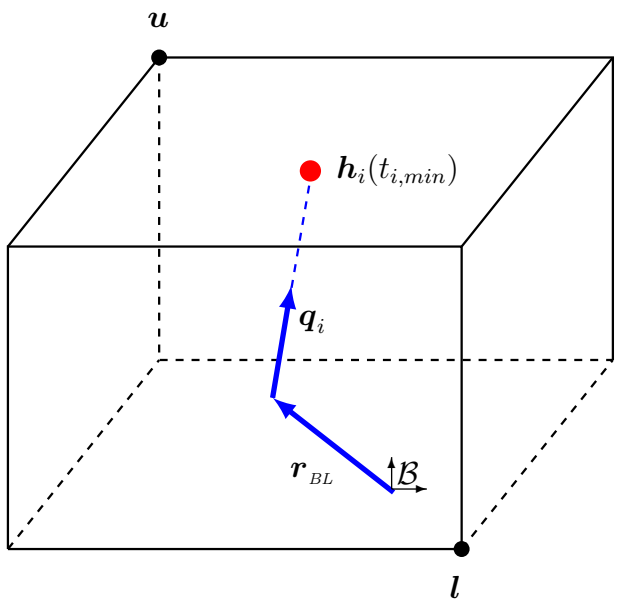


Figure 3.2: Intersections of a vector with the faces of a cuboid.

upper and lower boundaries of the cuboid are

$$\begin{aligned} \boldsymbol{u} &= \begin{bmatrix} \frac{\Delta x}{2} & \frac{\Delta y}{2} & \Delta z \end{bmatrix}^{\mathrm{T}} \\ \boldsymbol{l} &= \begin{bmatrix} -\frac{\Delta x}{2} & -\frac{\Delta y}{2} & 0 \end{bmatrix}^{\mathrm{T}} \end{aligned} \tag{3.17}$$

respectively. Δx, Δy and Δz are defined in Table 3.1. ${}^{B}\boldsymbol{r}_{BL}$ is the laser scanner position with respect to the inertial coordinate system $\underset{\rightarrow}{\mathcal{B}}$ while $\boldsymbol{q}_i$ is the vector of the i-th laser beam. The intersection of the laser beam with the reflector is represented by the first intersection of $\boldsymbol{q}_i$ with the boundary while the other intersections are neglected. The computation of the intersections is described in the next paragraph.

According to Nocedal and Wright [2006] (Section 16.6, Gradient Projection Method), the intersection of a vector and the faces of a cuboid is computed by

$$ {}^{B}\boldsymbol{h}_i = {}^{B}\boldsymbol{r}_{BL} + t_{i,min}\,{}^{B}\boldsymbol{q}_i \quad , \quad i \in \mathcal{I}_M. \tag{3.18} $$

The first intersection of i laser beam with the reflector occurs when

$$ t_{i,min} = \min\left\{ \boldsymbol{t}_{ij} \right\} \quad , \quad j = 1, 2, 3. \tag{3.19} $$

j component of $\boldsymbol{t}_{ij}$ for i laser beam is computed by

$$ t_{ij} = \begin{cases} \dfrac{u_j - {}^{B}r_{BLj}}{q_{ij}} & \text{if} \quad q_{ij} > 0 \\ \dfrac{l_j - {}^{B}r_{BLj}}{q_{ij}} & \text{if} \quad q_{ij} < 0 \\ \infty & \text{otherwise} \end{cases} . \tag{3.20} $$

Note that $0 < t_{i,min} \leq t_{max}$. t_{max} is given in Table 2.1.

Eq. 3.18 is only valid if the end effector is stationary while the laser scanner completes a scan/measurement. When the velocity effect of the end effector mentioned in subsection 3.1.2 is taken into account, then ${}^{B}\boldsymbol{r}_{BL}$ is changing over time. To avoid misconception, a new notation ${}^{B}\boldsymbol{r}_{ci}$, where $i \in \mathcal{I}_M$, is introduced to represent the laser scanner position vector with respect to $\underset{\rightarrow}{\mathcal{B}}$ instead of ${}^{B}\boldsymbol{r}_{BL}$. Suppose I_C is the index set of all indices of the scan number $I_C = \{0, 1, 2, ..., n_s\}$. The laser scanner position with respect to $\underset{\rightarrow}{\mathcal{B}}$ over time is given by

$$ {}^{B}\boldsymbol{r}_{ci} = \begin{cases} \boldsymbol{r}_p + {}^{B}\boldsymbol{r}_{PL} + \boldsymbol{v}_e\left(\tau_{inc} + \tau_t\right) & \text{if} \quad i = 1 \\ \boldsymbol{r}_p + {}^{B}\boldsymbol{r}_{PL} + \boldsymbol{v}_e\tau_{inc} & \text{otherwise} \end{cases} . \tag{3.21} $$

where $\boldsymbol{r}_p$ is the previous position of the end effector. The intersection point of the laser beam vector with the reflector is computed with Eq. 3.18, but ${}^{B}\boldsymbol{r}_{BL}$ is replaced with ${}^{B}\boldsymbol{r}_{ci}$.

Eq. 3.18 to Eq. 3.21 are intended to imitate the measurement data from the laser scanner. Since the velocity effect is taken into account, the measurement data during the platform while stationary and while in motion are different. The desired

measurement data that would be ideal for the further process is when the platform is stationary. To reduce the velocity effect on the measurement data during the platform motion, a compensation algorithm is developed in the following section.

3.2 Compensation of the velocity effect

In order to reduce if not eliminate the influence of the platform velocity (velocity effect) on the measurement data, the length of the laser beam must be compensated by shifting the actual position of the laser beam to its origin. Suppose $\boldsymbol{r}_1$ in Fig. 3.3 is the desired measurement result from the laser scanner in Cartesian coordinates.

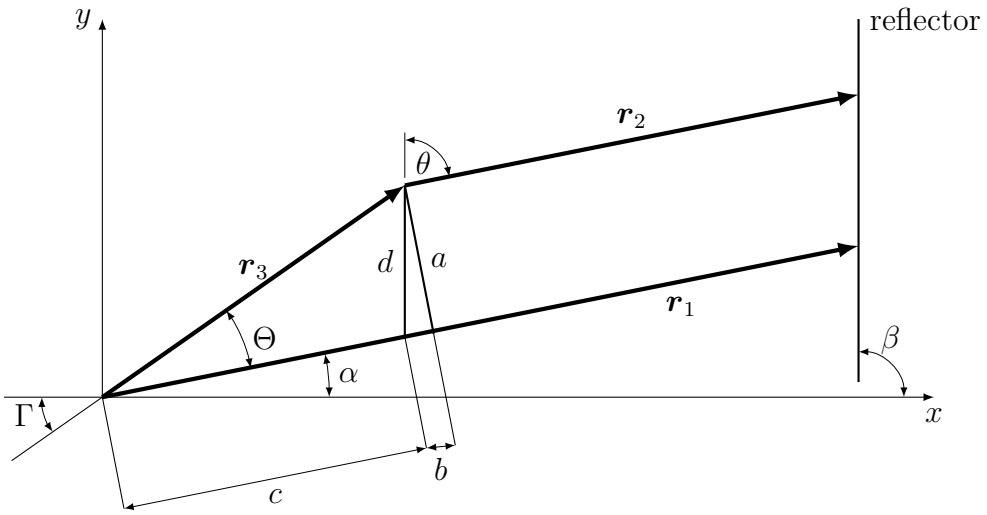


Figure 3.3: Compensation of the velocity effect.

Due to the velocity effect, the origin of $\boldsymbol{r}_1$ is shifted by

$$\boldsymbol{r}_3 = \boldsymbol{v}_e \tau_{inc}, \tag{3.22}$$

making the actual measurement result be $\boldsymbol{r}_2$. If the rotation about the z-axis is assumed constant during a scan, then the angle between $\boldsymbol{r}_2$ and the x-axis (denoted as α) is identical with the angle between $\boldsymbol{r}_1$ and the x-axis. Thus, the angle between $\boldsymbol{r}_3$ and $\boldsymbol{r}_1$ is obtained from

$$\Theta = \Gamma - \alpha, \tag{3.23}$$

where Γ is the angle between $\boldsymbol{r}_3$ and the x-axis. Then, the perpendicular distance between $\boldsymbol{r}_2$ and $\boldsymbol{r}_1$ is given by

$$a = \|\boldsymbol{r}_3\|_2 \sin\Theta. \tag{3.24}$$

According to Fig. 3.3, the magnitude of $\boldsymbol{r}_1$ is obtained by

$$\|\boldsymbol{r}_1\|_2 = \|\boldsymbol{r}_2\|_2 + c. \tag{3.25}$$

Since the line d is parallel with the reflector, the angle θ is computed by

$$\theta = \beta - \alpha, \tag{3.26}$$

and β is obtained from the previous measurement. The length b is computed by

$$b = \frac{a}{\tan\theta}, \tag{3.27}$$

where a is perpendicular distance from $\boldsymbol{r}_1$ to $\boldsymbol{r}_2$, and

$$b + c = \|\boldsymbol{r}_3\|_2 \cos\Theta. \tag{3.28}$$

Eq. 3.28 can be rewritten as

$$c = \|\boldsymbol{r}_3\|_2 \cos\Theta - b. \tag{3.29}$$

Substituting Eq. 3.29 into Eq. 3.25 yields the compensated beam length.

3.3 Reflector Model

Suppose CABLAR is setup as follows. The reflectors with pattern as depicted in Fig. 3.1 are mechanically fixed on the CABLAR frame. The laser scanner is mounted in the end effector with the same orientation and satisfies the right-hand rule for a system of axes. If the laser scanner operates with an aperture angle of 180 ° within the space surrounded by the reflectors at the upper and sides, the measurement data will have the form of a rectangular shape without a border at the bottom side. The proper extraction of the *normal parametrization*[1] from three borders[2] of a measurement result yields the distances from the laser scanner coordinate system $\mathcal{L}$ to all of the reflectors. Furthermore, since the normal parametrizations of the reflectors in the yz-plane are known, the y- and z-components of the platform position vector ${}^{B}\boldsymbol{r}_{BL}$ can be calculated easily. But the x-component, which is also important in this research, is not obtainable.

The reflector's pattern should be improved in order to generate unique measurement data proportional to the end-effector motion along the x-axis and the end-effector rotation about the z-axis. To fulfill that requirement, a new reflector

[1] The perpendicular distance from the line to the origin and its angle named according to Duda and Hart [1972].

[2] The borders considered as straight lines.

design with a transverse piece pattern on the upper side was introduced, as illustrated in Fig. 3.4. The upper side is chosen because this side is always scanned

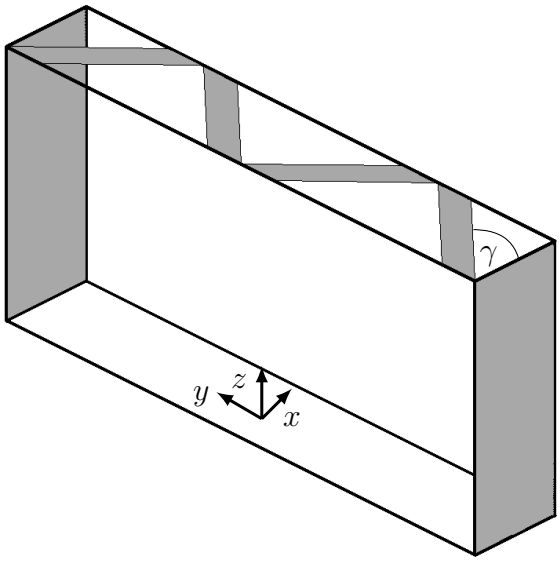


Figure 3.4: Special pattern of reflector design.

completely by the laser scanner regardless of the position of the end effector inside the working space. Algorithm 3.1 describes the step to design the reflector at the

Algorithm 3.1 Step to design the reflector arrangement

1: Divide the upper-plane into several small areas considering that the angle between the diagonal line and the side γ should be $30\,^\circ \leq \gamma \leq 60\,^\circ$.
2: Specify the width of the reflector. It should be no more than $\frac{1}{3}$ of the small area length.
3: The reflector tip must intersect with the frame corner at the upper side of the robot frame as depicted in Fig. 3.5 (e.g. the origin of $\boldsymbol{P}_i$) .

upper-side and generates the position vector $\boldsymbol{P}_i = [x, y]^T \in \mathbb{R}^2$ and its direction vector $\boldsymbol{Q}_i = [x, y]^T \in \mathbb{R}^2$ as depicted in Fig. 3.5, where $i = \{1, 2, 3..., n_l\}$. n_l is the number of all reflector edges, in this case, 4 reflectors multiplied with 2 edges for each reflector.

For this case, n_l is 8. $\boldsymbol{P}_1$ is specified by

$$\boldsymbol{P}_1 = \begin{cases} [l_x, u_y]^T & \text{for} \quad r_y > 0 \\ \\ [l_x, l_y]^T & \text{otherwise} \end{cases} . \tag{3.30}$$

where l_x, l_y, u_y and r_y are the x-component of $\boldsymbol{l}$, the y-component of $\boldsymbol{l}$, the y-component of $\boldsymbol{u}$, and the y-component of $\boldsymbol{r}$, respectively. Moreover, from $\boldsymbol{P}_2$ to

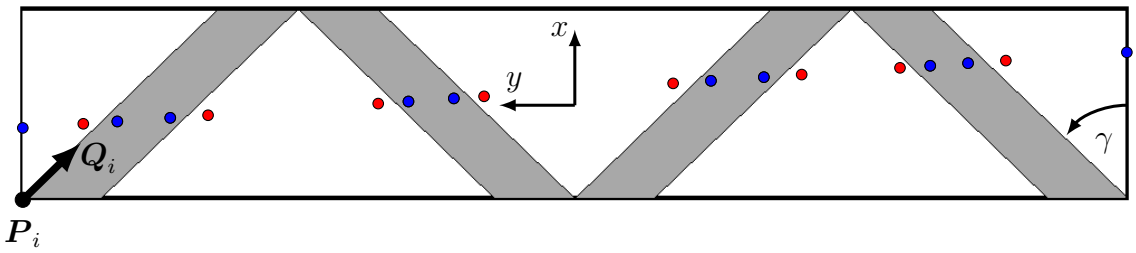


Figure 3.5: Top view of robot frame.

$\boldsymbol{P}_{n_l}$ are not specified here but are computed by employing Algorithm 3.1. In other words, $\boldsymbol{P}_2$–$\boldsymbol{P}_{n_l}$ gives the position of the reflector at the robot frame. Meanwhile, due to the inertial system of the cable robot, the usual quadrant of the Cartesian coordinate system is rotated 90° counter clockwise about the z-axis, as depicted in Fig. 3.6. Then, $\boldsymbol{Q}_i$ is computed by

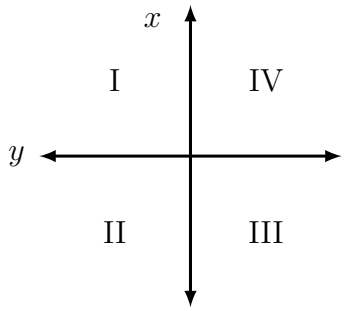


Figure 3.6: Quadrant of Cartesian coordinate system.

$$\boldsymbol{Q}_i = \begin{cases} \begin{bmatrix} \cos(\gamma) \\ \sin(\gamma) \end{bmatrix} & \text{for } \boldsymbol{Q}_i \text{ in quadrant I} \\ \\ \begin{bmatrix} \cos(-\gamma) \\ \sin(-\gamma) \end{bmatrix} & \text{for } \boldsymbol{Q}_i \text{ in quadrant IV} \end{cases} . \tag{3.31}$$

This modification of the reflector influences the intersection points/measurement data that correspond to the reflector at the upper side. In the next step, not all intersection points are considered to be processed. Only the measurement results that represent the distance from the laser scanner's origin to the particular reflectors are processed in the next step, while the remaining ones are neglected. The particular reflectors are the reflectors at the left and right sides and the transverse

reflector at the upper side. Geometrically, each reflector is considered as a plane and bounded horizontally by two straight lines with the linear equation

$$\zeta = m\xi + c, \tag{3.32}$$

where

$$\zeta = \begin{cases} z-\text{axis} & \text{for} \quad xz-\text{plane} \\ y-\text{axis} & \text{for} \quad xy-\text{plane} \end{cases}, \tag{3.33}$$

and

$$\xi = \begin{cases} x-\text{axis} & \text{for} \quad xz-\text{plane} \\ x-\text{axis} & \text{for} \quad xy-\text{plane} \end{cases}. \tag{3.34}$$

m and c are the slope and intercept, respectively. The slope of the lower and upper boundaries of each reflector is calculated by

$$m = \tan\gamma. \tag{3.35}$$

where

$$\gamma = \begin{cases} 0 & \text{for} \quad \text{reflector at left and right side} \\ \neq 0 & \text{for} \quad \text{reflector at upper side} \end{cases}. \tag{3.36}$$

The constant of Eq. 3.36 is known from the construction data, where the lower and upper boundaries of the reflector at the left and right sides are parallel with the abscissa (x-axis). Substituting Eq. 3.36 into Eq. 3.32 yields

$$\zeta = \begin{cases} c & \text{for} \quad \text{reflector at left and right side} \\ \tan(\gamma)\,\xi + c & \text{for} \quad \text{reflector at upper side} \end{cases}. \tag{3.37}$$

where

$$c = \begin{cases} l_z & \text{for} \quad \text{lower boundary of the reflector at left and right side} \\ u_z & \text{for} \quad \text{upper boundary of the reflector at left and right side} \\ P_{iy} & \text{for} \quad \text{all boundaries of the reflector upper side} \end{cases}. \tag{3.38}$$

P_{iy} denotes the y-component of $\boldsymbol{P}_i$.

An intersection point $\boldsymbol{h}_i$ is located on the particular reflector if the following condition is satisfied:

$$l_z \leq h_{iz} \leq u_z \tag{3.39}$$

for these reflectors at the left and right sides and

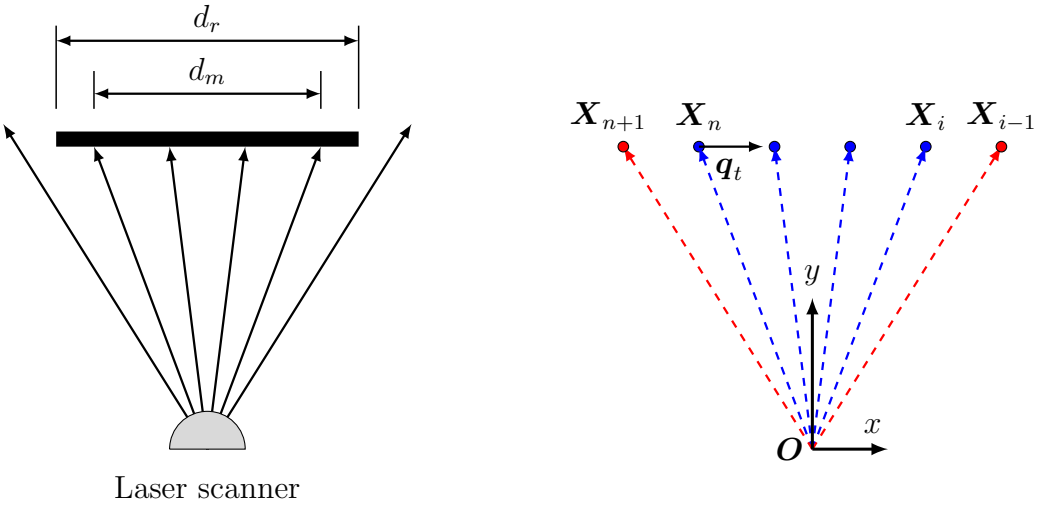
$$\tan(\gamma)\ h_{kx} + P_{ky} \leq h_{iy} \leq \tan(\gamma)\ h_{kx} + P_{(k+1)y} \tag{3.40}$$

for the reflector at the upper side. The subscript k and $(k+1)$ represent the left and right edge of a reflector[3] respectively. The desired measurement result is selected by

$$ {}^{B}\boldsymbol{h}_{v} = \begin{cases} {}^{B}\boldsymbol{h}_{i} & \text{if} \quad \text{Eq. 3.39} \vee \text{Eq. 3.40} \quad \text{satisfied} \\ [\,] & \text{otherwise} \end{cases} . \tag{3.41} $$

3.4 Measurement deviation due to the angular resolution of the laser scanner

Suppose the laser scanner is used to measure the length of a flat object, as depicted in Fig. 3.7.a. The line starting from the origin (laser scanner) to the object repre-



a. Measurement model

b. Expected result

Figure 3.7: Object detection using laser scanner.

sents the laser beam. The measurement result without regard for the systematic error of the sensor is illustrated as the blue-dashed vector ($\boldsymbol{X}_i$ to $\boldsymbol{X}_n$) in Fig. 3.7.b. Due to the non-continuity of the laser beam, the tips of the object do not intersect with the laser beam. In consequence, the real length of the object d_r can not be calculated but the minimum length can be calculated by

$$ d_m = \|\boldsymbol{X}_i - \boldsymbol{X}_n\|_2. \tag{3.42} $$

[3]The left and right sides are seen from the top view as depicted in Fig. 3.5.

The minimum length d_m is less than or equal to the real length d_r, or this can be written in mathematical form as

$$d_m \leq d_r. \tag{3.43}$$

To calculate the maximum length of the object d_u, the previous point before the first intersection and the posterior point after the last intersection of the laser beam with the object ($\boldsymbol{X}_{i-1}$ and $\boldsymbol{X}_{n+1}$) must be known. Those vectors are calculated by the following procedures:

1. $\boldsymbol{X}_{i-1}$ is obtained from the intersection between the line having vector equations $\boldsymbol{X}_n + \boldsymbol{q}_t$ with $\boldsymbol{O} + \boldsymbol{q}_{i-1}$, and

2. $\boldsymbol{X}_{n+1}$ is obtained from the intersection between the line having vector equations $\boldsymbol{X}_n + \boldsymbol{q}_t$ with $\boldsymbol{O} + \boldsymbol{q}_{n+1}$.

The gradient of the measured object is calculated by

$$\boldsymbol{q}_t = \frac{\boldsymbol{X}_i - \boldsymbol{X}_n}{d_m}, \tag{3.44}$$

with position vector

$$\boldsymbol{x}_p = \forall \boldsymbol{X} : \boldsymbol{X} = \{\boldsymbol{X}_i, \boldsymbol{X}_{i+1}, ..., \boldsymbol{X}_n\}. \tag{3.45}$$

Since the angular resolution of the laser scanner is constant, the direction vector of the laser beam is obtained by rotating the adjacent direction vector. For instance, $\boldsymbol{q}_i$ is rotated by θ[4] about the z-axis to obtain the unit vector of $\boldsymbol{X}_{i-1}$ as follows:

$$\boldsymbol{q}_{i-1} = \boldsymbol{q}_i \boldsymbol{R}, \tag{3.46}$$

where

$$\boldsymbol{q}_i = \frac{\boldsymbol{X}_i}{\|\boldsymbol{X}_i\|_2} \tag{3.47}$$

and

$$\boldsymbol{R}(\theta) = \begin{cases} \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} & \text{for } \theta = \Delta\Theta \\ \\ \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} & \text{for } \theta = -\Delta\Theta \end{cases}. \tag{3.48}$$

The same method is applied to calculate the unit vector of $\boldsymbol{X}_{n+1}$, which is denoted by $\boldsymbol{q}_{n+1}$, by replacing $\boldsymbol{X}_i$ with $\boldsymbol{X}_n$.

[4] θ for instance the angular resolution of the laser scanner.

By now, all the parameters needed to calculate these intersection points have been obtained and the computation of $\boldsymbol{X}_{i-1}$ and $\boldsymbol{X}_{n+1}$ can be performed. Appendix A describes how to calculate the intersection point of two vectors.

Finally the upper limit of the object length d_u is the distance between two vectors, which is calculated from the output of the step above

$$d_u = \|\boldsymbol{X}_{n+1} - \boldsymbol{X}_i\|_2. \tag{3.49}$$

Thereafter the upper limit is given in Eq. 3.43 and yields the following inequalities for the object's length

$$d_m \leq d_r \leq d_u. \tag{3.50}$$

3.5 Determination of the x-component of the end-effector's position

To extract the x-component of the platform position vector from the measurement data, a mathematical modeling based on the special reflector design is proposed in this section. It is started by introducing a set of perfect measurement results[5] of n_r pieces of reflectors at the upper side, which are indicated by the blue dots in Fig. 3.8. The corners and the normal parametrization of its measurement result

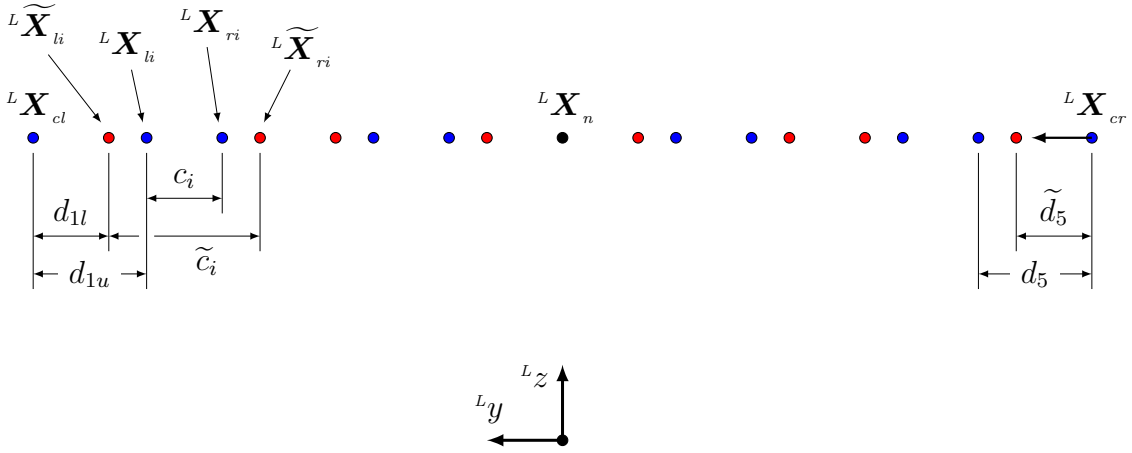


Figure 3.8: Desired measurement result.

in the Cartesian coordinate system are also included in Fig. 3.8 as the demand of mathematical modeling. The method of calculating the corner will be explained in the next chapter. The notations ${}^L\boldsymbol{X}_{cl}$ and ${}^L\boldsymbol{X}_{cr}$ stand for the vector at the left

[5] Without systematic error of the laser scanner.

and right corners of the robot frame respectively, while $\boldsymbol{X}_n$ is the vector where the direction vector of normal parametrization of the measurement point intersects with the line generated from the measurement points. All vectors consist of y- and z-components ${}^{L}\boldsymbol{X} = [{}^{L}y, {}^{L}z]^{\mathrm{T}}$ since the laser scanner scan is two-dimensional. The subscripts l, r and n stand for left, right side and normal parametrization, respectively. Although many points are generated by the laser scanner for each reflector, only the first and the last point on the reflector are selected to represent a reflector for the further step. In Fig. 3.8, the blue dots between the two red dots are the first and last points. It should be noted that the terms 'first and last point' are interchangeable, depend on the reference. To simplify, a pair of points which represents a reflector is denoted by ${}^{L}\boldsymbol{X}_{li}$ and ${}^{L}\boldsymbol{X}_{ri}$, where $i \in \mathcal{I}_{\mathcal{R}}$ and $\mathcal{I}_{\mathcal{R}} = \{1, 2, \cdots, n_r\}$.

In Section 3.4, the disadvantage of the use of a laser scanner to measure the length of a flat object has been described. The inequality in Eq. 3.50 is proposed to define the length of each object. A similar idea is now applied to define the width of each measured reflector. The lower bound for the width of each reflector is computed by

$$c_i \;=\; \| {}^{L}\boldsymbol{X}_{ri} - {}^{L}\boldsymbol{X}_{li} \|_2 \tag{3.51}$$

and displayed in Fig. 3.8. To compute the upper bound for the width, the points adjacent to $\boldsymbol{X}_{li}$ and $\boldsymbol{X}_{ri}$, namely $\widetilde{\boldsymbol{X}}_{li}$ and $\widetilde{\boldsymbol{X}}_{ri}$, must be known. Section 3.4 describes the idea and how to compute these adjacent points. The following equation determines the upper bound of the reflector width

$$\widetilde{c}_i \;=\; \| {}^{L}\widetilde{\boldsymbol{X}}_{ri} - {}^{L}\widetilde{\boldsymbol{X}}_{li} \|_2 \tag{3.52}$$

as depicted in Fig. 3.8.

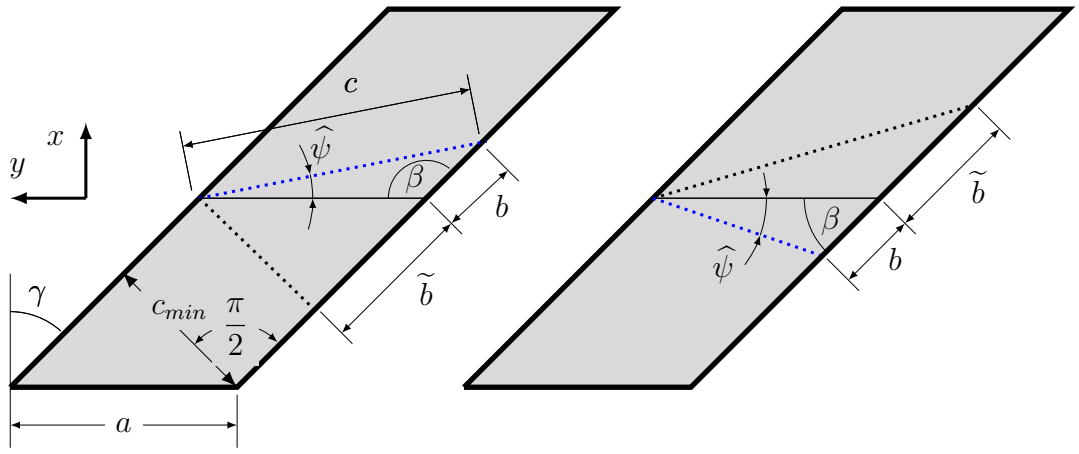
c_i and $\widetilde{c}_i$ are important for approximating the rotation of the platform about the z-axis (ψ) with respect to the robot's inertial system. The blue-dotted line in Fig. 3.9 is an illustration of the intersection between the laser scanner and a reflector when the platform rotates about the z-axis in the positive direction (Fig. 3.9.a) and the negative direction (Fig. 3.9.b). The parameter c is calculated by Eq. 3.51 and Eq. 3.52) while the angle

$$\beta = \begin{cases} \dfrac{\pi}{2} + \gamma & \text{if } c \geq a \\ \\ \dfrac{\pi}{2} - \gamma & \text{if } c < a \end{cases} . \tag{3.53}$$

The angle γ and the constant a are known from the design data.

According to the cosine equation, b is calculated by

$$b^2 = a^2 + c^2 - 2ac\cos\widehat{\psi}. \tag{3.54}$$



a. Case I, $c \geq a$ b. Case II, $c < a$

Figure 3.9: Yaw angle determination.

Rearranging Eq. 3.54 yields

$$\widehat{\psi} = \arccos\left(\frac{a^2 - b^2 + c^2}{2ac}\right) \tag{3.55}$$

which is the estimated magnitude of the platform rotation about the z-axis with respect to the inertial system while its direction remains unknown.

The unknown parameter b of Eq. 3.55 is calculated by the cosine law

$$c^2 = a^2 + b^2 - 2ab\cos\beta. \tag{3.56}$$

Rearranging Eq. 3.56 yields a quadratic equation

$$b^2 - (2a\cos\beta)\, b + \left(a^2 - c^2\right) = 0. \tag{3.57}$$

Since all the coefficients in Eq. 3.57 are known, b is calculated by the quadratic formula for finding the roots as follows

$$b_{1,2} = \frac{2a\cos\beta \pm \sqrt{(-2a\cos\beta)^2 - 4\left(a^2 - c^2\right)}}{2} \tag{3.58}$$

or simplified as

$$b_{1,2} = \underbrace{a\cos\beta}_{A} \pm \underbrace{\sqrt{c^2 - a^2\sin^2\beta}}_{B}. \tag{3.59}$$

Since $b_{1,2}$ are real distances, then B needs to be a real number. B is a real number if

$$c \geq a \sin \beta. \tag{3.60}$$

In the real system, c is a minimum when the collinear points of the measurement result are perpendicular with the edge of the reflector, as depicted in Fig. 3.9. In that event, c is equal to the closest edge distance of a reflector, or computed by

$$c_{min} = a \cos \gamma. \tag{3.61}$$

In terms of β, Eq. 3.61 is rewritten by substituting Eq. 3.53 as follows

$$c_{min} = \begin{cases} a \cos\left(\beta - \dfrac{\pi}{2}\right) & \text{if } c \geq a \\ \\ a \cos\left(\dfrac{\pi}{2} - \beta\right) & \text{if } c < a \end{cases}. \tag{3.62}$$

Applying a trigonometric identity simplifies Eq. 3.62 to

$$c_{min} = a \sin \beta. \tag{3.63}$$

Since Eq. 3.63 satisfies Eq. 3.60, it can be concluded that the B part of Eq. 3.59 is always a real number.

After determining the part B, the roots of Eq. 3.59 are specified as follows

$$\begin{aligned} b_1 &= A + B \\ b_2 &= A - B \end{aligned} \tag{3.64}$$

and the inequalities of the part A can be specified from Eq. 3.53 as follows

$$A \begin{cases} < 0 & \text{if } c \geq a \\ \\ > 0 & \text{if } c < a \end{cases}. \tag{3.65}$$

Substituting Eq. 3.65 into Eq. 3.64 yields the inequalities for the roots

$$|b_1| \begin{cases} > |b_2| & \text{if } c > a \\ \\ < |b_2| & \text{if } c \leq a \end{cases}. \tag{3.66}$$

Two new notations are introduced to differenciate the solutions according to their lengths

$$\begin{matrix} b = \min\left(|b_1|, |b_2|\right) \\ \widetilde{b} = \max\left(|b_1|, |b_2|\right) \end{matrix}. \tag{3.67}$$

The projection of the solutions (b and $\widetilde{b}$) on the reflector as depicted in Fig. 3.9 simplifies the determination of the desired solution from among the two solutions. The blue and black dotted-line in Fig. 3.9 has a length of c, but the angle between the lines and the ${}^{B}\boldsymbol{z}$-axis are different. Since the end-effector rotation is assumed small[6] then b is the desired solution to compute the rotation of the platform about the ${}^{B}\boldsymbol{z}$-axis ($\widehat{\psi}$) by inserting b into Eq. 3.55.

b can also be computed by using the signum function as follows

$$b = A - \operatorname{sgn}(A)B \tag{3.68}$$

where sgn is the signum function.

Substituting b of Eq. 3.68 into Eq. 3.55 yields the magnitude of the angle of rotation about the z-axis ($\widehat{\psi}$), but the direction remains unknown. The direction is identified by comparing c with a, illustrated in Fig. 3.9. If i starts from the left, as depicted in Fig. 3.5, then the magnitude and the direction of the rotation are specified by

$$\psi = \begin{cases} \widehat{\psi} & \text{if } \left[(c_i \vee c_{i+2}) \geq a\right] \vee \left[(c_{i+1} \vee c_{i+3}) < a\right] \\ -\widehat{\psi} & \text{else} \end{cases}. \tag{3.69}$$

Since the reflectors consist of four transverse pieces, then four values of ψ are obtained. Only one among the four is chosen for the further step. To choose the best from among the four ψ, c must be classified according to its reflector direction. Two notations are introduced for each set of data, as follows

$$\begin{matrix} \check{c} = \left\{c_i, c_{i+2}, \cdots, c_{n_r-1}\right\} \\ \check{c} = \left\{c_{i+1}, c_{i+3}, \cdots, c_{n_r}\right\}. \end{matrix} \tag{3.70}$$

Now, the best c must be selected for each set of data by considering a criteria, which now will be discussed.

Section 3.4 explains the use of the laser scanner to estimate the length of a flat object. If the laser scanner is used to estimate the width of two or more flat objects which have same dimensions and are placed with the same orientation, then the maximum estimated length must be the best answer. This idea is applied to choose

[6]The end-effector motion is designed only for planar motion

the best value from two sets of data in Eq. 3.70. The best number is the maximum of the data set, or written mathematically,

$$\begin{array}{l} \check{c}_{max} = \max\left(\check{c}\right) \\ \breve{c}_{max} = \max\left(\breve{c}\right) \end{array}. \tag{3.71}$$

It should be noted that the indices of the selected measurement result (i) of Eq. 3.52 must be stored in order to calculate its upper bound.

Both the number from Eq. 3.71 and its upper bound[7] are then used to computed two pairs of ψ, by substituting Eq. 3.71 into Eqs 3.57–3.69. The lower bound of $\check{c}_{max}$ and its upper bound turn out to be $\check{\psi}_b$ and $\check{\psi}_u$ while the lower bound of $\breve{c}_{max}$ and its upper bound are $\breve{\psi}_b$ and $\breve{\psi}_u$. Then, the rotation of the platform is specified by two intervals

$$\check{\psi}_b \leq \psi \leq \check{\psi}_u \tag{3.72}$$

and

$$\breve{\psi}_b \leq \psi \leq \breve{\psi}_u. \tag{3.73}$$

The subscripts b and u stand for lower and upper bound, respectively.

Even if both intervals are eligible for ψ approximation, only the interval with the shortest length is considered. For the purpose of approximation, in the in the next step the interval ψ will be discretized into a certain number of subintervals. With the same increment value, the data set with the shortest interval has less data than the other one. The small amount of data also saves computer memory and requires less computational effort in the calculation. The statement in this paragraph is formulated mathematically in the following paragraph.

Suppose $\check{\psi}_s = \left\{\check{\psi}_b, \check{\psi}_u\right\}$, $\breve{\psi}_s = \left\{\breve{\psi}_b, \breve{\psi}_u\right\}$ and $\psi_s = \{\psi_{sb}, \psi_{su}\}$. Then

$$\psi_s = \begin{cases} \check{\psi}_s & \text{if } \left(\check{\psi}_u - \check{\psi}_b\right) \leq \left(\breve{\psi}_u - \breve{\psi}_b\right) \\ \\ \breve{\psi}_s & \text{else} \end{cases}, \tag{3.74}$$

where ψ_s, ψ_{sb}, ψ_{su} are the shortest interval, lower- and upper bounds of ψ, respectively. ψ_s is required to determine the x-component of the platform pose, where its mathematical modeling starts from the following paragraph.

Suppose the blue-line in Fig. 3.10 is a projection of the measurement data on the upper-side of the robot frame, where the laser beams intersect with the platform. Although the laser measurement points are discontinuous, it is assumed continuous

[7] the upper bound is specified from the indices of the selected measurement result, discussed in the paragraph above and early in this section

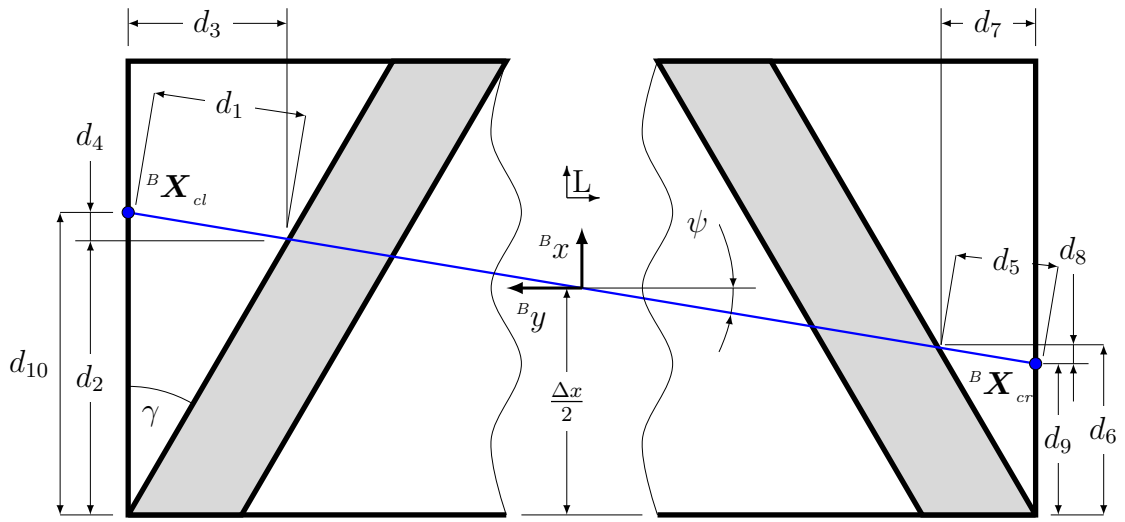


Figure 3.10: Projection of ideal measurement data on the upper side reflector.

to simplify the modeling, and the discontinuity will be considered later on. The γ and ψ are known from the design data. The magnitude of the blue-line is computed from the measurement data which is depicted in Fig. 3.8.

In terms of the inertial system $\underline{\mathcal{L}}$, the blue-line starts from the corner ${}^B\boldsymbol{X}_{cl} = [{}^Bx_{cl}, {}^By_{cl}, {}^Bz_{cl}]^{\mathrm{T}}$ to the corner ${}^B\boldsymbol{X}_{cr} = [{}^Bx_{cr}, {}^By_{cr}, {}^Bz_{cr}]^{\mathrm{T}}$ or vice versa. The y- and z-components of ${}^B\boldsymbol{X}_{cl}$ and ${}^B\boldsymbol{X}_{cr}$ are known from the construction data (where the reflectors are mounted) while the x-component is going to be calculated. The x-component can be computed from the measurement data either at the left or right side of the reflector. However, the calculation on the side which is closer to the end effector will generate a more accurate result. The model is developed for the rotation ψ in the negative direction, but is also applicable for a positive direction.

The aim of this modeling is to obtain the length from the x-component of ${}^B\boldsymbol{X}_{cl}$ or ${}^B\boldsymbol{X}_{cr}$ to the bottom border of the rectangle[8] illustrated in Fig. 3.10. The lengths are denoted by d_{10} and d_9 for the left side and right side, respectively. These lengths are calculated by the following equations

$$\begin{aligned} d_9 &= d_6 - d_8 \\ d_{10} &= d_2 + d_4. \end{aligned} \tag{3.75}$$

[8]The upper side of the robot frame.

The parameters in Eq. 3.75 are given in the following equations

$$\begin{aligned} d_6 &= \frac{d_7}{\tan\gamma} && (3.76)\\ d_8 &= d_5 \sin\psi && (3.77)\\ d_2 &= \frac{d_3}{\tan\gamma} && (3.78)\\ d_4 &= d_1 \sin\psi && (3.79)\\ d_3 &= d_1 \cos\psi && (3.80)\\ d_7 &= d_5 \cos\psi. && (3.81) \end{aligned}$$

d_1 and d_5 are obtained from the measurements. Since the laser beams are assumed continuous, then Eq. 3.75 has a unique solution. However, the laser beams are not continuous, as mentioned in Section 3.4, so d_1 and d_5 are expressed as intervals as follows

$$\begin{aligned} d_{1b} \le d_1 \le d_{1u} \\ d_{5b} \le d_5 \le d_{5u} \end{aligned} \qquad (3.82)$$

where

$$\begin{aligned} d_{1b} &= \| {}^{L}\boldsymbol{X}_{cl} - {}^{L}\widetilde{\boldsymbol{X}}_{l1} \|_2 \\ d_{1u} &= \| {}^{L}\boldsymbol{X}_{cl} - {}^{L}\boldsymbol{X}_{l1} \|_2 \end{aligned} \qquad (3.83)$$

and

$$\begin{aligned} d_{5b} &= \| {}^{L}\boldsymbol{X}_{cr} - {}^{L}\widetilde{\boldsymbol{X}}_{r(n_r)} \|_2 && (3.84)\\ d_{5u} &= \| {}^{L}\boldsymbol{X}_{cr} - {}^{L}\boldsymbol{X}_{r(n_r)} \|_2. && (3.85) \end{aligned}$$

Substituting Eq. 3.83 into Eq. 3.80 and Eq. 3.85 into Eq. 3.81 yields

$$\begin{aligned} d_{3u} &= d_{1u} \cos\left(\min|\psi_s|\right) && (3.86)\\ d_{3b} &= d_{1b} \cos\left(\max|\psi_s|\right) && (3.87) \end{aligned}$$

for the left side and

$$\begin{aligned} d_{7u} &= d_{5u} \cos\left(\min|\psi_s|\right) && (3.88)\\ d_{7b} &= d_{5b} \cos\left(\max|\psi_s|\right) && (3.89) \end{aligned}$$

for the right side. The new distance from x-component of ${}^{B}\boldsymbol{X}_{cl}$ or ${}^{B}\boldsymbol{X}_{cr}$ to the bottom border of the rectangle[9] is computed by

$$\begin{aligned} d_{10u} &= d_{2u} - d_{1u}\cos\psi_{sl} && (3.90)\\ d_{10b} &= d_{2bl} - d_{1b}\cos\psi_{su} && (3.91) \end{aligned}$$

[9] upper side of the robot frame

for the left side and

$$d_{9u} = d_{6u} + d_{5u}\cos\psi_{su} \tag{3.92}$$
$$d_{9b} = d_{6b} + d_{5b}\cos\psi_{sl} \tag{3.93}$$

for the right side and also presented as an inequality in term of the inertial coordinate system $\mathcal{B}$ as

$$\widehat{d}_{10b} \leq {}^{B}x_{cl} \leq \widehat{d}_{10u} \tag{3.94}$$

for the left side of the robot frame and

$$\widehat{d}_{9b} \leq {}^{B}x_{cr} \leq \widehat{d}_{9u} \tag{3.95}$$

for the right side, where

$$\begin{aligned}
\widehat{d}_{10b} &= d_{10b} - \frac{\Delta x}{2} \\
\widehat{d}_{10u} &= d_{10u} - \frac{\Delta x}{2} \\
\widehat{d}_{9b} &= d_{9b} - \frac{\Delta x}{2} \\
\widehat{d}_{9u} &= d_{9u} - \frac{\Delta x}{2}
\end{aligned} \tag{3.96}$$

By now, the proposed modeling above has provided the rotation ψ and x-component of ${}^{B}\boldsymbol{X}_{cl}$ or ${}^{B}\boldsymbol{X}_{cr}$ expressed as an interval, or stated in other words, that the solutions lie in the intervals. Since the unique solutions of ψ and x-component are not possible to be achieved due to the discontinuity of the laser beam, an approach is proposed now to determine both parameters, which is described in the following.

The left side of the robot frame at the upper side with some vectors are depicted in Fig. 3.11. ${}^{B}\boldsymbol{X}_{cl}$ coincides with ${}^{L}\boldsymbol{X}_{cl}$ and lies on the same straight blue line with ${}^{L}\widetilde{\boldsymbol{X}}_{li}$, ${}^{L}\boldsymbol{X}_{li}$, ${}^{L}\boldsymbol{X}_{ri}$, and ${}^{L}\widetilde{\boldsymbol{X}}_{ri}$, where $i = \{1, 2, \cdots, n_r\}$ and n_r is the number of reflectors. The angle between the blue line and ${}^{L}y$ is denoted by ψ. The intersections of the blue line with each reflector edge are denoted by ${}^{B}\boldsymbol{u}_{li}$ and ${}^{B}\boldsymbol{u}_{ri}$ for the left and right edge respectively. These ***ideal points*** would be generated by the laser scanner if its beams were continuous.

The reflector edges are represented by position vectors ${}^{B}\boldsymbol{p}_i$, ${}^{B}\boldsymbol{r}_i$ and direction vectors ${}^{B}\boldsymbol{q}_i$, ${}^{B}\boldsymbol{s}_i$, which are known from the construction data. If ${}^{B}\boldsymbol{X}_{cl}$, ${}^{L}\boldsymbol{X}_{cl}$, ${}^{L}\widetilde{\boldsymbol{X}}_{li}$, ${}^{L}\boldsymbol{X}_{li}$, ${}^{L}\boldsymbol{X}_{ri}$, ${}^{L}\widetilde{\boldsymbol{X}}_{ri}$ and the rotation angle ψ are obtained from the measurement, then ${}^{B}\boldsymbol{u}_{li}$ and ${}^{B}\boldsymbol{u}_{ri}$ can be obtained by calculating the intersection of ${}^{B}\boldsymbol{q}_i$ with ${}^{B}\boldsymbol{v}$ and the intersection of ${}^{B}\boldsymbol{s}_i$ with ${}^{B}\boldsymbol{v}$ respectively, where

$${}^{B}\boldsymbol{v} = [\cos\psi, \sin\psi, \Delta z]^{\mathrm{T}} \tag{3.97}$$

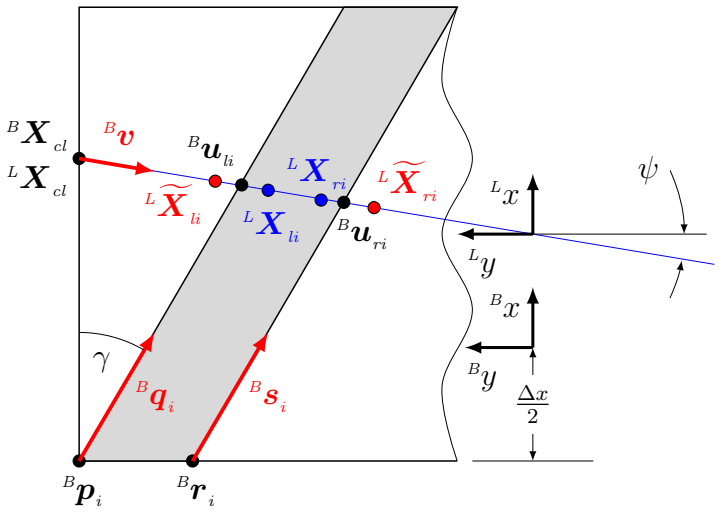


Figure 3.11: Measurement data on the upper reflector.

for the computation at the upper left of the robot frame and

$$ {}^{B}\boldsymbol{v} = \left[\cos\left(\pi + \psi\right), \sin\left(\pi + \psi\right), \Delta z\right]^{\mathrm{T}} \tag{3.98} $$

for the upper right. Appendix A describes how to compute the intersection of two vectors.

An equally important step in this section is the computation of the distance from the corner to the measured and ideal points, formulated as follows:

$$ \begin{aligned} h_{li} &= \left\| {}^{B}\boldsymbol{u}_{li} - {}^{B}\boldsymbol{X}_{cl} \right\|_2 \\ h_{ri} &= \left\| {}^{B}\boldsymbol{u}_{ri} - {}^{B}\boldsymbol{X}_{cl} \right\|_2 \\ \widetilde{t}_{li} &= \left\| {}^{L}\widetilde{\boldsymbol{X}}_{li} - {}^{L}\boldsymbol{X}_{cl} \right\|_2 \\ t_{li} &= \left\| {}^{L}\boldsymbol{X}_{li} - {}^{L}\boldsymbol{X}_{cl} \right\|_2 \\ \widetilde{t}_{ri} &= \left\| {}^{L}\widetilde{\boldsymbol{X}}_{ri} - {}^{L}\boldsymbol{X}_{cl} \right\|_2 \\ t_{ri} &= \left\| {}^{L}\boldsymbol{X}_{ri} - {}^{L}\boldsymbol{X}_{cl} \right\|_2 \end{aligned} . \tag{3.99} $$

According to Fig. 3.11 and Eq. 3.99, it can be stated that the distance from the corner ${}^{B}\boldsymbol{X}_{cl}$ to the ideal point (${}^{B}\boldsymbol{u}_{li}$ or ${}^{B}\boldsymbol{u}_{ri}$) is between the distance from the corner ${}^{B}\boldsymbol{X}_{cl}$ to the neighbouring points of an ideal point, or formulated mathematically as

$$ \widetilde{t}_{li} \leq h_{li} \leq t_{li} \tag{3.100} $$

for the left edge and

$$ \widetilde{t}_{ri} \leq h_{ri} \leq t_{ri} \tag{3.101} $$

for the right edge.

Eqs. 3.97–3.101 are the main idea to estimate ${}^{B}x_{cl}$ and the rotation angle ψ. Suppose any ψ and ${}^{B}x_{cl}$ in the interval of Eq. 3.74 and Eq. 3.94 are substituted into Eq. 3.97 and Eq. 3.99. If Eq. 3.100 as well as Eq. 3.101 is satisfied, a vote value of 1 is given. The vote values are summed for all distances from the corner ${}^{B}\boldsymbol{X}_{cl}$ to the ideal point[10]. For instance, 8 is the maximum vote if the number of reflectors (n_r) is 4 and Eq. 3.100 as well as Eq. 3.101 are satisfied for all i. The number of votes, ψ and ${}^{B}x_{cl}$, are stored for further processing.

Eq. 3.74 and Eq. 3.95 must be specified within a reasonable range[11] for the next step, such that:

$$ {}^{B}x_{cl} = \left\{ \widehat{d}_{10b}, \left(\widehat{d}_{10b} + d_j \right), \left(\widehat{d}_{10b} + 2d_j \right), \cdots, \widehat{d}_{10u} \right\} \tag{3.102} $$

and

$$ \psi_s = \left\{ \psi_{sb}, \left(\psi_{sb} + \psi_j \right), \left(\psi_{sb} + 2\psi_j \right), \cdots, \psi_{su} \right\}, \tag{3.103} $$

where d_j and ψ_j are constants that specify the increments of ψ and ${}^{B}x_{cl}$. Furthermore, all values in Eq. 3.102 and Eq. 3.103 are combined and substituted into Eq. 3.97 and Eq. 3.99. Then all vote values are recorded together with ψ and ${}^{B}x_{cl}$. The combinations of ψ and ${}^{B}x_{cl}$ which has the highest vote are the desired solution. The solution is still in the form of an interval, but the length between the start- and end-point of the solution is shorter than before. To obtain a single value, mathematical statistics such as the arithmetic mean or median can be employed.

The method for the determination of the yaw angle proposed in this section is based on the assumption that the laser beams always intersect with all the reflectors at the left, right and upper sides of the robot frame as depicted in Fig. 3.11. That means that the rotation of the platform determined by the proposed method in this section has a limitation, which is explained in the following section.

3.6 Limitation of yaw angle

In this section, the measurement limitation of the platform rotation about the z-axis is discussed. The yaw angle can be determined if the laser beam intersects all the sides of the reflectors. The top view of the intersection is illustrated in Fig. 3.12 According to Fig. 3.12, the yaw angle is computed by

$$ \psi = \tan \frac{0.5 w_r}{r_y}, \tag{3.104} $$

[10] there are 8 ideal points for 4 reflectors

[11] depending on the desired accuracy

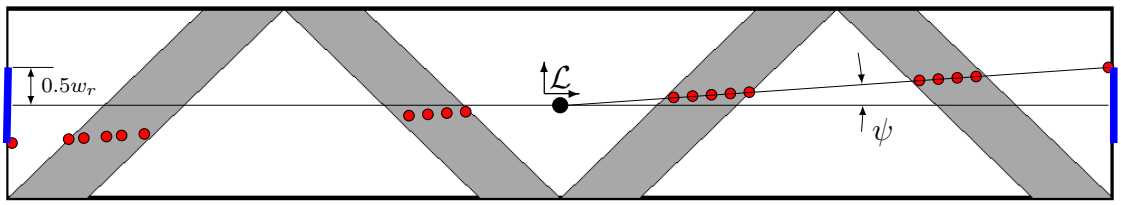


Figure 3.12: Upper view of robot frame with laser beam.

where w_r is the width of the reflector at the left and right side while r_y is the y-component of the platform position vector. If the assumptions:

- pitch angle of the platform orientation is zero,
- x-component of platform position is zero for all,

are valid, then the yaw angle function of the y-component of the platform position is depicted in Fig. 3.13.

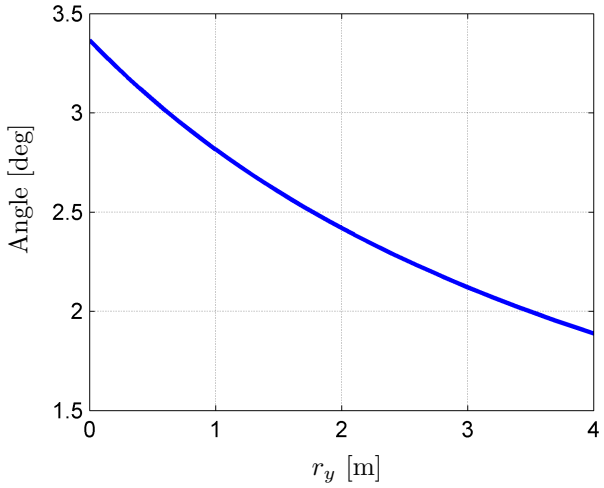


Figure 3.13: Limit of yaw angle.

3.7 Concluding remarks

The kinematic model of the robot including the laser scanner properties was established. The model is valid for the platform in stationary and during the motion. To compensate the measurement result during the platform motion, the mathematical model is proposed. Since the laser scanner measured the distance to the object in 2D, only two among three translational components of platform pose can be determined. A special pattern of reflector design was proposed to obtain the

rest translational component and one rotational component (Yaw angle). A mathematical model based on the reflector design and considering the angular resolution of the laser scanner was developed.

CHAPTER 4

Straight Line Extraction and Strategy to Determine Platform Pose

It is very often in the engineering field that the collected data has random errors and so-called *outliers*, for several reasons, such as uncertainty or noise. In this chapter, the focus is on the outliers. An outlier is also known as a *gross error* that can occur due to several reasons, such as the following [Barnett, 1978]:

1. measurement error,
2. execution faults,
3. intrinsic variability.

Instead of rejection, Ben-Gal [2005] proposed detecting outliers because some important information could be held by an outlier. In this work, one step towards obtaining the end-effector position is the extraction of straight line parameters from a data set which can be from simulation or measurement. To obtain a coherent analysis, a set of data corresponding to a straight line, the following steps are employed:

1. detecting the outlier,
2. rejecting the outlier and generating new data by curve fitting.

Detecting an outlier by one of the methods in [Ben-Gal, 2005] followed by a straight line fitting by polynomial regression, for instance the least squares method, could be a reliable solution. But, handling the measurement data containing several straight lines requires more effort, particularly to detect an outlier, because many

points form its membership. Instead of performing the steps above separately with different approaches, an estimation method that is able to handle both steps is worth considering.

In Tab. 4.1, an experimental result of six popular algorithms to extract the straight line parameters for the application in mobile robotic and computer vision have been investigated experimentally by Nguyen et al. [2005] in order to compare their performance. A 2D laser scanner mounted on an indoor mobile robot was used to collect 100 measurement data (scans) from an 80 mx50 m office area. The benchmark of his result is given in Table 4.1 with terminology:

- N: Number of points in input scan (722),
- S: Number of line segments extracted (7 on average, depending on algorithm),
- N_f: Sliding window size for *Line-Regression*,
- $N.Trials$: Number of trials for RANSAC (1000),
- NC, NC: Number of columns, rows respectively for the HT accumulator array (NC=401, NR=671 for resolution r_{res}=1 cm, α_{res}=0.9 °),
- $N1$, $N2$: Number of trials and convergence iterations, respectively, for EM (N1=50, N2=200).

and

$$\begin{aligned} TruePos &= \frac{N.Matches}{N.TrueLines} \\ FalsePos &= \frac{N.LineExByAlgo - N.Matches}{N.LineExByAlgo}, \end{aligned} \tag{4.1}$$

where $N.LineExByAlgo$, $N.Matches$ and $N.TrueLines$ are the number of lines extracted by an algorithm, the number of matches to true lines and the number of true lines, respectively.

Nguyen et al. [2005] conclude several points according to Table 4.1. Some of his conclusion related with this research are in the following:

- *Split-and-Merge*, *Incremental*, and *Line Regression* work faster than the others.
- *RANSAC* and *EM* result in higher *FalsePos* than the others.
- *RANSAC*, *HT*, and *EM+Clus* extract more precise lines, but have low frequency.
- In Overall, due to its speed and correctness, *Split-and-Merge* and *Incremental* are recommended for SLAM application.
- Application and implementation details influence the algorithm choice.

Table 4.1: Experimental results of the algorithm. Source: [Nguyen et al., 2005]

Algorithm	Complexity	Speed Hz	N.Lines	Correctness		Precision	
				TruePos	FalsePos	$\tau\Delta r$	$\tau\Delta\alpha$
				[%]	[%]	[cm]	[deg]
Split-Merge+Clust	$N \times \log N$	1470	641	86.0	8.9	1.95	0.74
Incremental	$S \times N^2$	344	561	77.8	5.9	2.04	0.72
Incremental + Clus		617	567	79.2	5.1	2.04	0.76
Line Regression (LR)	$N \times N_f$	364	577	76.4	10.1	1.99	0.80
LR + Clus		384	562	75.8	8.4	1.97	0.79
RANSAC	$S \times N \times N.Trials$	29	749	75.6	31.5	1.68	0.77
RANSAC + Clus		93	547	70.7	12.2	1.37	0.70
Hough Transform (HT)	$S \times N \times NC + S \times NR \times NC$	8	825	82.0	32.5	1.63	0.76
HT + Clus		9	600	79.5	10.0	1.51	0.67
EM	$S \times N1 \times N2 \times N$	0.6	1153	78.6	53.7	2.09	0.97
EM + Clus		0.76	709	80.3	23.1	1.58	0.73

Meanwhile, the need for this thesis is a method which has the following properties:

- real-time capable,
- low memory use,
- simple programming,
- high precision.

None of the methods above fulfills completely these requirements. Alternatively, *RANSAC*, *HT*, *Split-and-Merge* and *Line Regression* are combined in order to meet these demands. In the following sections, these methods will be briefly explained.

4.1 Hough Transform

The Hough Transform (HT) was invented by Paul Hough in 1962 to detect straight lines in digitized images [Hough, 1962]. However, the generalized HT used today was invented by Duda and Hart [1972]. This technique is explained briefly in the following paragraphs.

Suppose the n collinear points shown in Fig. 4.1 (black points, without(x_p, y_p)) are

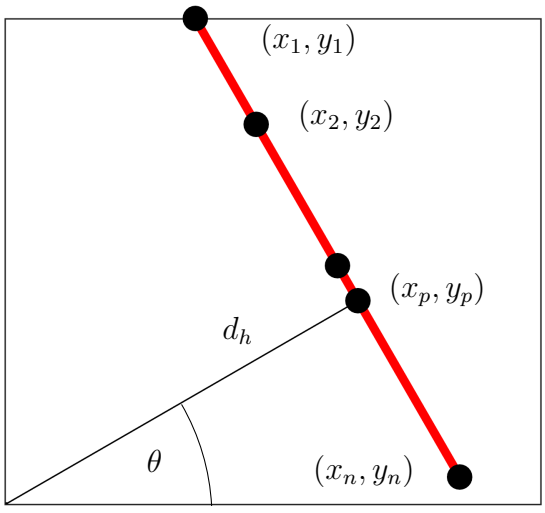


Figure 4.1: Straight line and its normal parametrization.

a set of measurement data. A straight line (red line) is fitted from the collinear points, where its normal parametrization is the angle (θ) between its normal and

the x-axis and the distance from the origin to its normal[1] (d_h). Then, the points in Fig. 4.1 are transformed into the parameter space[2] by the following equation

$$d_h = x_i \cos\theta + y_i \sin\theta, \tag{4.2}$$

for the points in Cartesian coordinate system or

$$d_h = l_i \cos\left(\alpha_i - \theta\right), \tag{4.3}$$

if polar coordinates are desired [Giesler et al., 1998]. l_i and α_i are the representation of (x_i, y_i) in polar coordinates. θ is given in a specific interval and $\{i \in \mathbb{Z} \mid 0 < i \leq n\}$. The computation complexity is derived by

$$c_c = n_p n_t, \tag{4.4}$$

where n_p and n_t are the number of data points, and the number of θ quantization/discretization, respectively.

Eq. 4.2 transforms points in Fig. 4.1 the into $d_h - \theta$ plane or parameter space, as depicted in Fig. 4.2. Four sinusoidal curves are shown in Fig. 4.2. In other words,

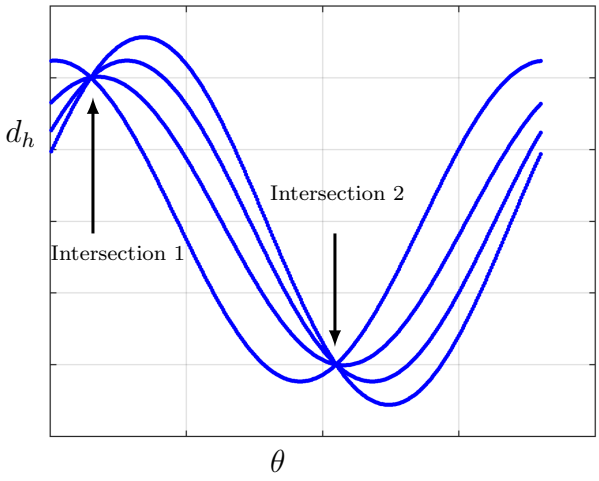


Figure 4.2: Sinusoidal curve in parameter space.

a curve corresponds with a point in Fig. 4.1. Since these points are collinear, all the sinusoidal curves pass through the same point in the parameter space. Duda and Hart [1972] summarized interesting properties of the point-to-curve transformation:

1. A point in the picture plane corresponds to a sinusoidal curve in the parameter plane.

[1] also known as the shortest distance to the origin
[2] also known as a sinusoidal curve

2. A point in the parameter plane corresponds to a straight line in the picture plane.
3. Points lying on the same straight line in the picture plane correspond to curves through a common point in the parameter plane.
4. Points lying on the same curve in the parameter plane correspond to lines through the same point in the picture plane.

The normal parametrization of these points is obtained from the coordinates of the intersection points on the $\theta - d_h$ plane. If θ is limited only to the interval [0,π], the sinusoidal curves will intersect at one point. However, theta is expanded to the interval [0,2π] in order to get two intersection points for a specific reason, explained later in this section. The second intersection point is the mirror image of the first one.

Duda and Hart [1972] proposed also the transformation of the points from the Cartesian or polar coordinate system into the so-called accumulator array. The accumulator array is a matrix to record the vote from each point of each curve in Fig. 4.2. The size of the matrix is specified by the quantization of θ and d_h, where $0 \leq \theta \leq \pi$ (if π is maximum) and $-R \leq d_h \leq R$, where R is defined as the size of the retina and d_h is from Eq. 4.2 or Eq. 4.3. The R must be greater than $max\,(d_h)$ to make sure the vote can be stored in the accumulator array. A vote is added to every cell in the accumulator array corresponding to every point in all the sinusoidal curves. The vote is accumulated when a cell receives more than one vote. As a result, Fig. 4.3 shows the accumulator array of the point in Fig. 4.1. The cells correspond with the intersection points in Fig. 4.2 receiving more votes. A threshold must be defined to obtain the cell which has votes more than the threshold. This cell position within the accumulator array represents the normal parametrization.

Since the normal parametrization must be quantized, this leads to the drawbacks of this method: It is difficult to choose an appropriate grid size [Nguyen et al., 2005]. Increasing the resolution of the accumulator array would be a promising solution, but that is proportional to the computation cost [Duda and Hart, 1972, Illingworth and Kittler, 1988].

To overcome the second drawback above, Li et al. [1986] proposed another strategy as illustrated in Fig. 4.4. In order to reduce the accumulator array, the parameter space is divided into hypercubes in several steps, from a rough to a fine resolution. The size of the accumulator array is constant for each step. Then the HT is performed only on the hypercube which has more votes than the threshold. The idea is suitable to be adapted in this research work, but with an improvement. Before the improved idea is explained, the original idea is explained in more detail in the next paragraph.

Before the process is started, the threshold must be specified at the beginning. Suppose 8 lines pass through the same point in the 2D parameter space bounded

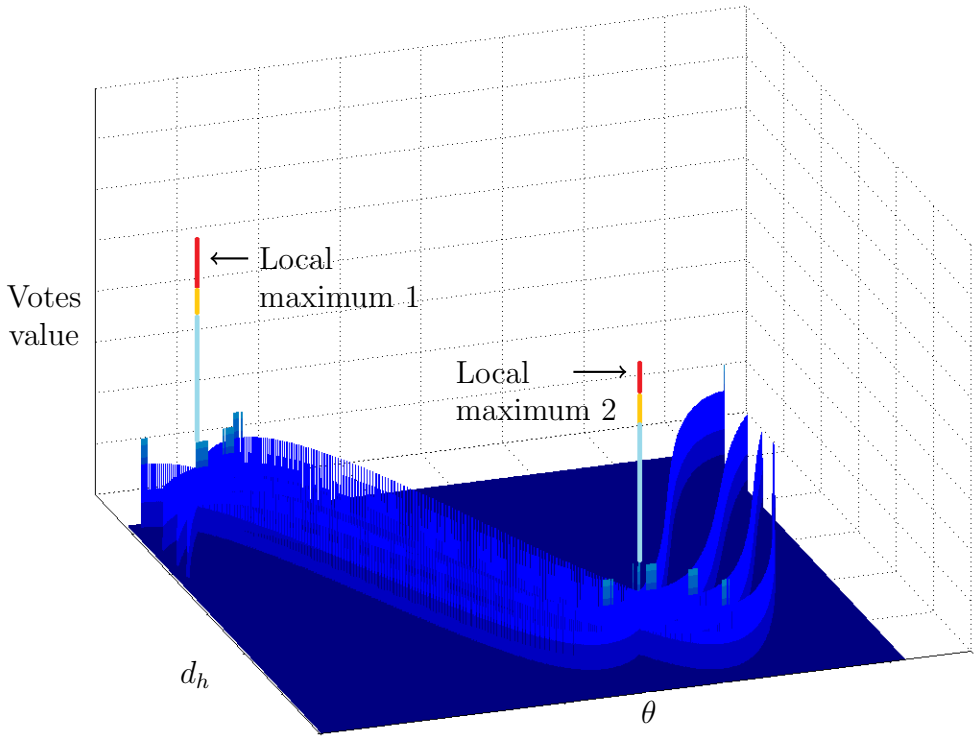
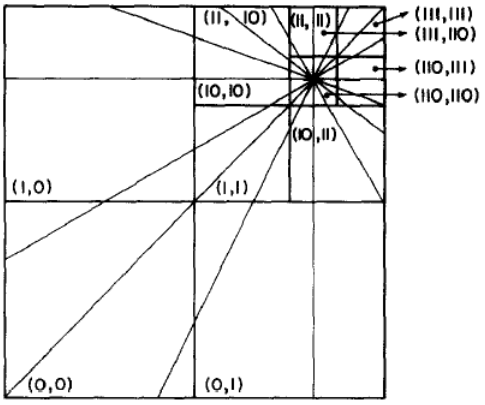


Figure 4.3: The accumulator array.



VOTE	K-TREE ADDRESS			
SIZE 1/2 x 1/2	(0,0)	(0,1)	(1,0)	(1,1)
	3	3	4	8
SIZE 1/4 x 1/4	(10,10)	(10,11)	(11,10)	(11,11)
	3	5	6	8
SIZE 1/8 x 1/8	(110,110)	(110,111)	(111,110)	(111,111)
	8	4	4	3

Figure 4.4: Example Fast Hough Transform process [Li et al., 1986].

by a unit square as depicted in Fig. 4.4. Then, a hypercube is divided into four hypercubes with the same sizes, named according to the node, as [0,0], [0,1], [1,0], and [1,1]. The hypercube containing more votes than the threshold is selected (hypercube [1,1]) and the remaining are neglected. The same process is repeated until the desired hypercube resolution is reached, for this case the hypercube [110,110]. With the same grid size of accumulator array, this method needs only $\frac{1}{64}$ size of accumulator array from classical HT. However, the iteration number and iteration time from rough until fine hypercube must be taken into account.

The idea above is improved for the application in this work. Instead of starting the division from rough until fine, a fine hypercube is applied directly to the particular position in parameter space. 'Particular position' means the position where the intersection point is located inside the fine hypercube, for instance, the area most near to the intersection point in Fig. 4.2. In comparison with the original idea in [Li et al., 1986], this modification is more efficient because the iterative process is shorter and finally reduces the computational time. However, this proposed idea is only applicable if θ and d_h can be estimated so that the fine hypercube can be placed in the desired position.

For the application in this work, the idea in the paragraph above is realizable since the platform motion always starts from a specific position, namely, the home position inside the CABLAR working space. This means the normal parametrizations of all reflectors at the initial position are known. The HT is performed only for a small specific area bounded by a rectangle, where its position is estimated by a prior[3] normal parametrization. An appropriate size of rectangle, which is determined according to the speed of the reflector, ensures that the intersection points are always inside the rectangle.

Without losing generality, a numerical example is shown to give an idea to the reader how significantly the modified HT decreases the accumulator array size and reduces the computational complexity. Consider n measurement points $\boldsymbol{h}_i \in \mathbb{R}^2$ where $\{i \in \mathbb{Z} \mid 0 < i \leq n\}$, depicted in Fig. 4.5. The collinear point within the green, black and red rectangles are the points of interest where their normal parametrization is going to be computed. All points in Fig. 4.5 are transformed into the $d_h - \theta$ plane (parameter space) by Eq. 4.3. θ is specified in the interval $\theta = \{0, \theta_{inc}, 2\theta_{inc}, 3\theta_{inc}, ..., 2\pi\}$, where θ_{inc} and π are, respectively, the increment of θ and the mathematical constant 3.14. θ_{inc} can be freely defined considering the precision of the parameter space. The computational results are depicted in Fig. 4.6. The relationship between the set of collinear points in Fig. 4.5 and their normal parametrizations in Fig. 4.6 is indicated by giving them the same colors.

The parameter space in Fig. 4.6 shows 10 intersection points, even though the number of collinear points set in Fig. 4.5 is four (the points in the green, red, and black rectangles and the remaining points). Usually, each intersection point is

[3] A word in the Latin language for 'earlier'.

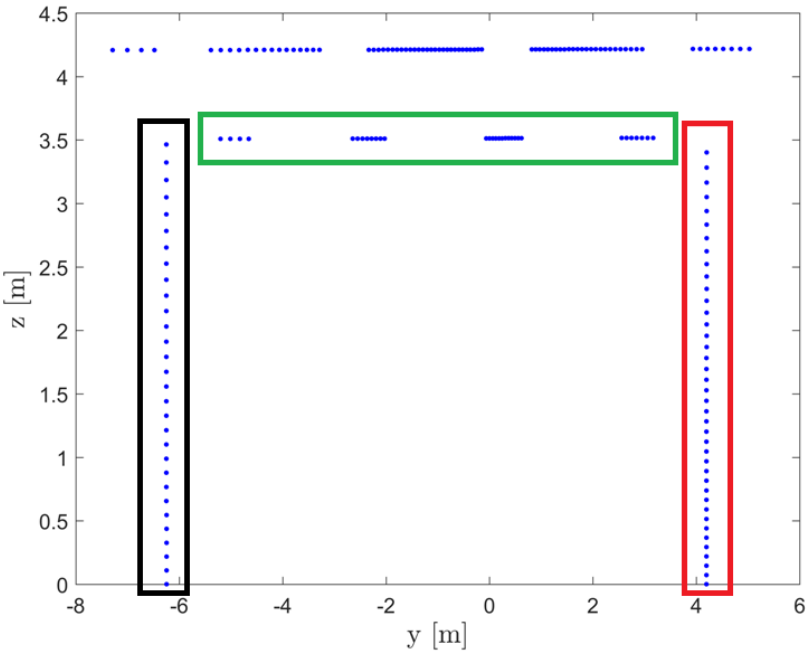


Figure 4.5: Measurement data from a scan.

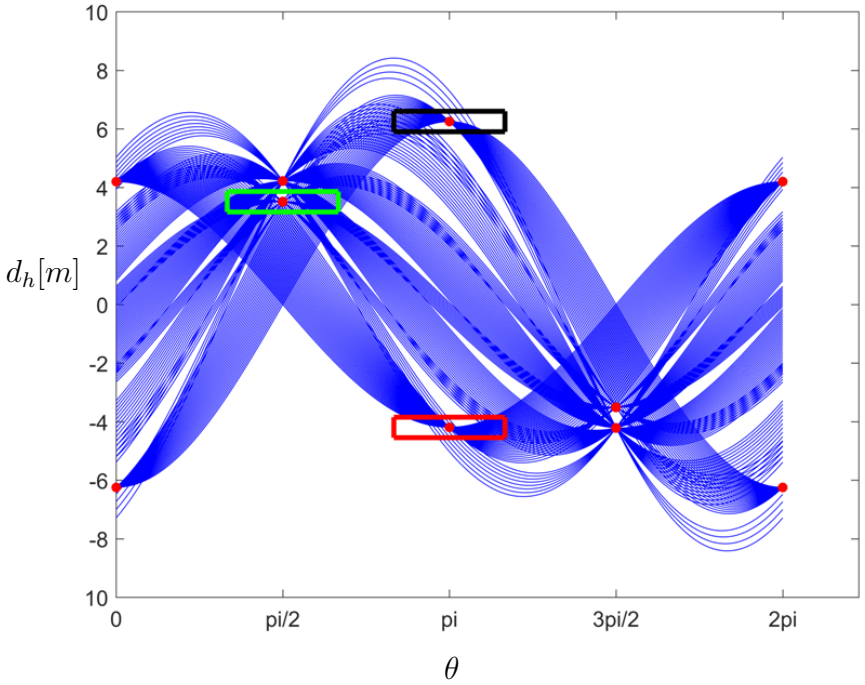


Figure 4.6: Parameter space.

replicated once in the interval $\{\theta \in \mathbb{R} \mid 0 < \theta \leq 2\pi\}$ in the parameter space, but since the θ component of the normal parametrization of the collinear points in the black and red rectangles lies on $\theta = 0$, the same intersection points exist on $\theta = 2\pi$. In other words, the intersection points on $\theta = 2\pi$ are the mirror images of the intersection points on $\theta = 0$. If the θ of these collinear points is less than zero, the intersection points appear only in the region $\pi \leq \theta \leq 2\pi$. For simplicity in the computer programming, the hypercube of the collinear point which has θ component close to 0° and 2 *pi* is placed on $\theta = \pi$ instead of $\theta = 0$.

Then, all votes correspond to the curve in the parameter space in Fig. 4.6 are stored in an accumulator array, as depicted in Fig. 4.7. The number of rows ($nrow$) and

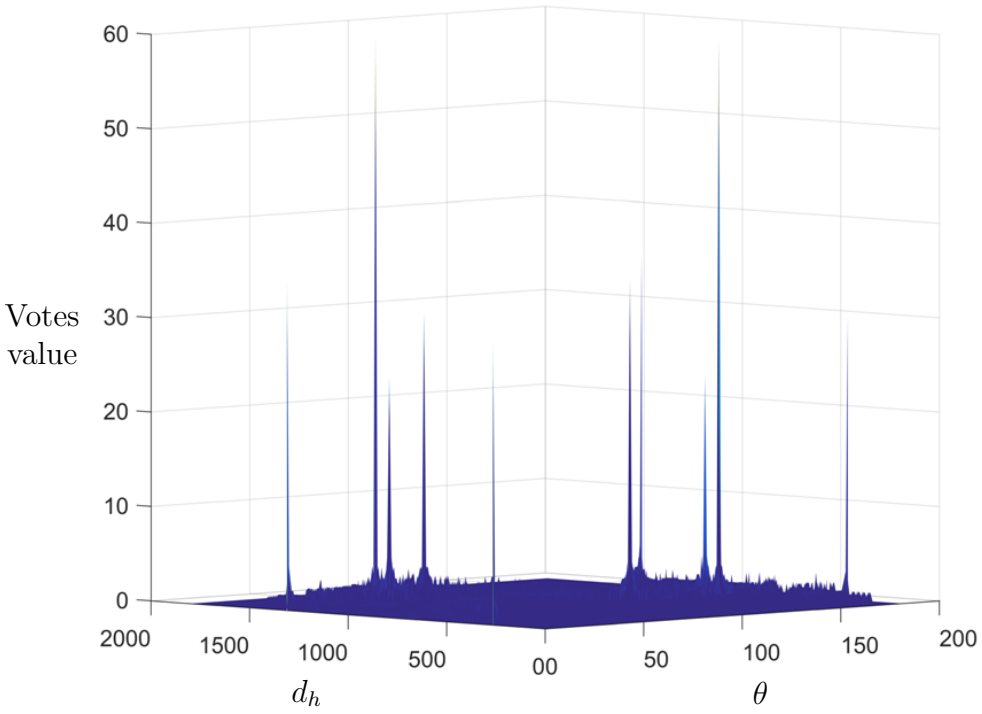


Figure 4.7: An accumulator array with different view point.

columns ($ncol$) are computed by

$$\begin{aligned} nrow &= 2d_{res}R, \\ ncol &= \frac{max\,(\theta)}{\theta_{inc}}, \end{aligned} \tag{4.5}$$

where R, and d_{res} are the retina size and resolution of the accumulator column. θ has the interval $\{\theta \in \mathbb{R} \mid 0 < \theta \leq 2\pi\}$. The constant 2 is obtained from the interval of retina size $\{R \in \mathbb{R}, d_h \in \mathbb{R} \mid -R \leq d_h \leq R\}$. Since the size of the accumulator array is always positive (greater than zero), the d_h interval can be rewritten as $0 \leq d_h \leq 2R$, where $R = max\,(l_i)$. After all votes are recorded in the parameter

space, the intersection points are obtained by selecting the cell containing votes exceeding the threshold.

An example of classical HT has been presented above. For comparison, the modified HT will be shown. Suppose the rotation of the end effector about the x-axis is limited to $\pm\,\theta_m$[4]. Then the interval θ can be rewritten as $\{\theta \in \mathbb{R} \mid (\theta_h - \theta_m) \leq \theta \leq (\theta_h - \theta_m)\}$ where θ_h is the normal parametrization component of a particular set of collinear points, obtained when the end effector is at the home position. The lower and upper boundary of the rectangle are specified by $\big(d_{h(k-1)} - d_m\big) \leq d_{h(k)} \leq \big(d_{h(k-1)} + d_m\big)$. d_m is computed by

$$d_m = \boldsymbol{v}_s \tau_c c_s, \tag{4.6}$$

where c_s is a scaling factor to keep $d_{h(k-1)}$ always inside the rectangle[5]. Here, the subscript k is introduced to distinguish the current and prior state. Fig. 4.8 illustrates the rectangle in the parameter space with its boundary. The full line

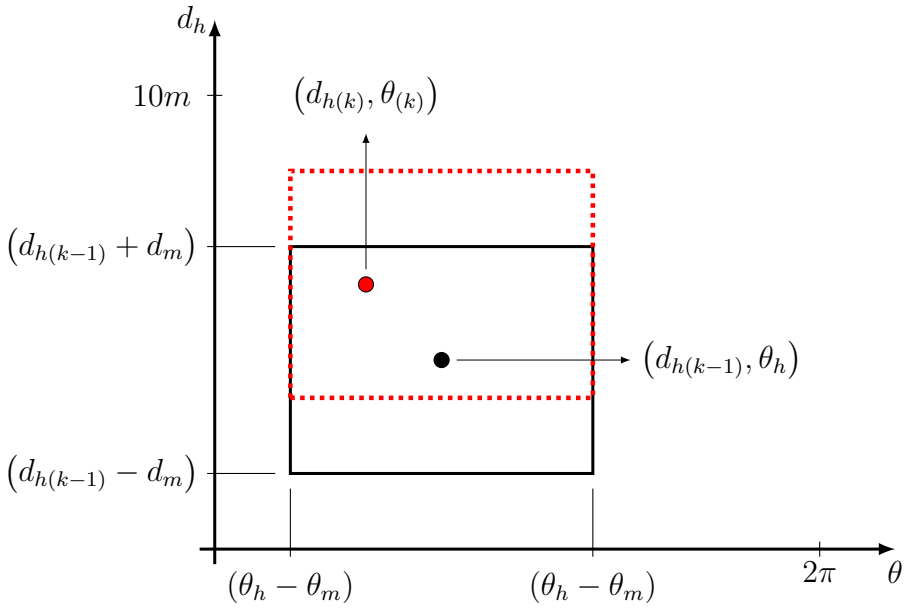


Figure 4.8: Fine size of rectangle in the parameter space.

rectangle represents the area of the current state where its vote is recorded in the accumulator array. Eq. 4.2 or Eq. 4.3 is performed with input from the data point in Fig. 4.5 with θ interval $\{\theta \in \mathbb{R} \mid (\theta_h - \theta_m) \leq \theta \leq (\theta_h - \theta_m)\}$. After all votes are recorded in the accumulator array, the peak is selected. Instead of finding the peaks in the accumulator array based on the threshold, the maximum value is considered as the peak. Consequently, the amount of the peak can possibly

[4] θ_m is obtained from experiment by measuring $\boldsymbol{R}_x(\varphi)$ of the end-effector using a sensor

[5] In an experiment, it could occur that the measurement result from the laser scanner for a certain scan is not received in a packet due to high data traffic. These data will be neglected.

decrease. Since the accumulator array is dimensionless, the indices of the largest value (ζ, ξ) are converted to the normal parametrization by

$$\theta = \text{round}\left((\theta_h - \theta_m) + \xi\theta_{inc}\right) \tag{4.7}$$

$$d_h = \frac{\text{round}\left(\left(d_{h(k-1)} - d_m\right) d_{res}\right) + \zeta}{d_{res}} \tag{4.8}$$

where 'round' means rounding the value to the nearest integer.

If the end effector is moving, the algebraic length $d_{h(k)} \neq d_{h(k-1)}$ while the angle could be $\theta_{(k)} \neq \theta_{(k-1)}$ or $\theta_{(k)} = \theta_{(k-1)}$ depends on the rotation $\boldsymbol{R}_x(\varphi)$. The red rectangle defines the new area to be calculated with the same process as before. The rectangle is always moving vertically to keep the prior algebraic length $d_{h(k-1)}$ always at the middle of the rectangle width.

By now, examples of the classical and the modified HT have been presented. To have a performance overview of both HTs, a comparison of the standard and modified HTs with parameters as in Table 4.2 is shown in Table 4.3. d_{res} scale 1:100

Table 4.2: Parameters of standard and modified HT

Parameter	Standard HT	Modified HT
Number of data (n)	181	181
Threshold	5	$max\,(H)$
θ_{inc}	2 °	2 °
$max\,(l_i)$	8.412 m	8.412 m
d_{res}	1:100	1:100

Table 4.3: Comparison of standard and modified HT

Parameter	Standard HT	Modified HT
$ncol$	181	46
$nrow$	1800	70
Complexity $(n \times ncol)$	32761	8326
Number of cell $(nrow \times ncol)$	325800	3220

means 1 m is quantized to be 100 columns in an accumulator row. Note must be taken that the modified HT is performed only to obtain the normal parametrization for one set of collinear points. The same calculation process must be repeated to obtain all desired normal parametrizations. In terms of complexity, since the

Table 4.4: Indices of peaks in standard HT accumulator array

No	row	column
1	275	1
2	1320	1
3	1251	46
	1252	46
4	1321	46
	1322	46
5	480	91
6	1525	91
7	478	136
	479	136
8	548	136
	549	136
9	275	181
10	1320	181

number of repetitions is three, the modified HT 24 % is lower than the standard HT. In terms of computer memory, the modified HT uses an extremely small amount of memory compared to the standard HT due to the decrease in the number of cells of the accumulator array to 1 %. This is a great benefit since memory use is one of big issues in real-time application.

The other interesting aspect to be discussed in this comparison is the method to specify the threshold value. As mentioned above, the threshold is usually defined at the beginning and is static for the whole simulation. Finding the peaks in the accumulator array based on the threshold implies possibly obtaining undesired peaks, which will be described in the following example. Table 4.4 shows the indices for all intersection points selected in the standard HT (threshold is defined at the beginning). 'No.' denotes the number of intersection point, while the row and column are the index numbers of the row and column, respectively. It can be seen that intersection points numbers 3, 4, 7, and 8 have more than one peak (vote values that exceed the threshold). The row index of those points are neighboring numbers and have the same column index.

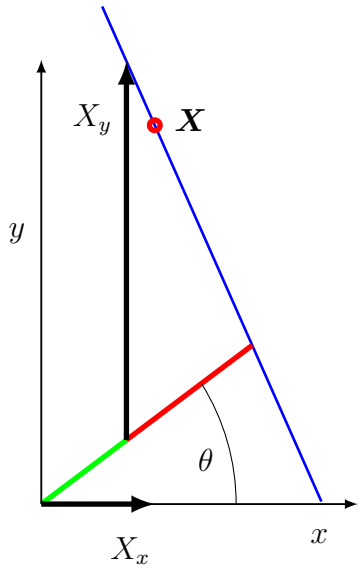


Figure 4.9: Point identification.

To observe the multi-peaks in more detail, a small part of the accumulator array which belongs to the green rectangle of the parameter space in Fig. 4.6 is displayed in Eq. 4.9.

$$\begin{matrix} 0 & 0 & 0 & 0 & 4 \\ 0 & 0 & 0 & 3 & 2 \\ 2 & 0 & 0 & 5 & 3 \\ 2 & 3 & \mathbf{24} & 4 & 1 \\ 2 & 5 & \mathbf{7} & 0 & 0 \\ 2 & 4 & 0 & 0 & 0 \\ 3 & 0 & 0 & 1 & 0 \end{matrix} \tag{4.9}$$

For instance, 6 is defined as the threshold and leads to obtaining two peaks (highlighted by the red color), although only one was desired. That was due to inappropriate quantization of the normal parametrization (d_{res} and θ_{inc}) as the size of the accumulator array and rounding in Eq. 4.7 to Eq. 4.8. For that reason the vote is spread to the neighboring cell in the accumulator array. Increasing the threshold can be a solution but not robust for the case where the peak values are close to each other. Nevertheless, this problem is solved by finding the peak by the maximum value instead of the threshold.

Another important task in HT is to find the points associated with the obtained straight line parameter or in other words, which point belongs to which line by reversing the HT process. Suppose d_{ht} and θ_{ht} are obtained by HT from collinear points in Fig. 4.1. Now, these normal parametrizations are re-transformed into the xy plane as a straight line, illustrated as the blue line in Fig.4.9. One can say that $\boldsymbol{X}$ (the red point) lies on the straight line by visual observation or mathematically by

$$|d_c - d_{ht}| \leq \epsilon \tag{4.10}$$

where

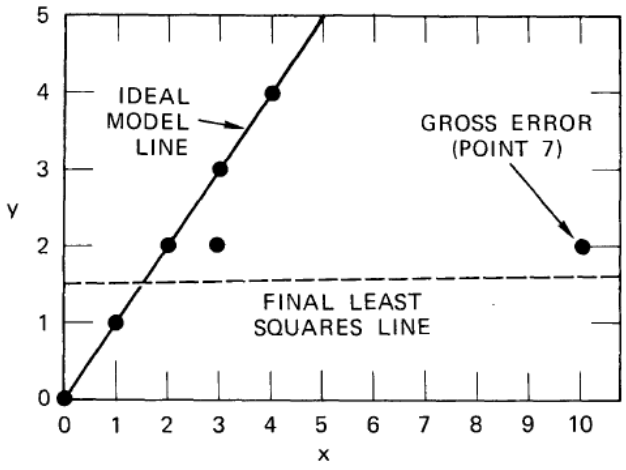
$$d_c = X_x \cos\theta_{ht} + X_y \sin\theta_{ht} \tag{4.11}$$

and ϵ, X_x, X_y are the defined error/tolerance, the x-, y-component of $\boldsymbol{X}$. Eq. 4.11 can be visualized as depicted in Fig. 4.9. The green line is the projection of X_x on the perpendicular distance of the blue line to the origin, which starts from origin and coincide with the x-axis. The end of the green line is the starting point of the projection X_y on the same perpendicular distance. If d_{ht} and θ_{ht} are the exact solution, then $d_c = d_{ht}$. But, due to round up error, an inappropriate size of the grid in the accumulator array, and the error function of its point from the ideal straight line, the exact solution can not be achieved. To decrease the sensitivity, ϵ is defined to allow a certain tolerance.

4.2 Random sample consensus

RANdom SAmple Consensus (RANSAC) gets a lot of attention and is widely accepted by researchers since its invention. Searching on the keyword RANSAC in Google scholar (on 05.08.2015) generated ca. 876,000 results, many more in comparison with Hough Transform (circa 80,500 results) and split-and-merge (279,000).

The main benefit or RANSAC is that it is simple and therefore suitable for real-time application, and it is able to handle data contaminated with outliers if its parameter is tuned appropriately. Fig. 4.10 illustrates the application of the least



POINT	x	y
1	0	0
2	1	1
3	2	2
4	3	2
5	3	3
6	4	4
7	10	2

Figure 4.10: Failure of least squares (and the "throwing out the worst residual" heuristic), to deal with an erroneous data point [Fischler and Bolles, 1981].

squares method (a classical technique) to estimate the ideal model line. A set of data with a gross error is shown in Fig. 4.10. Fitting the model line by least

squares yields the final least squares line (dashed-line) which is totally different compared to the ideal model line. The gross error has greatly influenced the final least squares line. According to that result, Fischler and Bolles [1981] prefers to detect the outlier and then reject the outlier by an internal mechanism in the RANSAC algorithm. Consequently, RANSAC is still applicable for the data set containing more than 50 % outlier [Zuliani, 2014].

Regardless of the many variations and applications of RANSAC, all such algorithms generally consist of two steps [Zuliani, 2014]:

1. **Hypothesize**. Model parameters are estimated from *minimal sample sets* (n_m) which are chosen randomly from the set of data.

2. **Test**. The whole data set is compared with reference to the model parameters. The elements belonging to the desired area are considered as the *consensus set* while the rest are rejected (outliers).

Both steps are repeated to achieve the best parameter, which is determined by having the largest consensus set. The process is terminated when the desired condition is achieved, such as the number of iterations.

Fischler and Bolles [1981] mentioned the parameters in the RANSAC method in the following

1. **Error tolerance**. Error tolerance specifies the distance from the instantiated model to the boundaries. It can be estimated either analytically or experimentally. Although each datum has a different error tolerance, a single value is sufficient to be defined in RANSAC with regard to quite high differences between all the errors tolerance and outliers.

2. **Number of trials**. The number of trials k to determine the ideal model parameters is given by

$$k = \frac{\log(1-z)}{\log(1-w^n)} \tag{4.12}$$

   where n, z, and w are the number of good data points, the probability of selecting at least one error-free sample of a set of n data points in k trials, and the fraction of inliers, respectively. Svoboda [2008] proposed an algorithm to compute the number of trials if w is unknown, which is often the case in a real problem. The pseudocode is given in Algorithm 4.1.

3. **Threshold distance**. The threshold must be large enough so that RANSAC can choose a significant amount of uncontaminated data in order to obtain the best model parameters. Moreover, the amount of uncontaminated data must be sufficient for the further processes, such as the smoothing process using least squares.

To have a better overview of the RANSAC algorithm, a numerical example is given. Consider $\boldsymbol{H}$, the set of numerical data in Table 4.5, with cardinality $\mid \boldsymbol{X} \mid = nX$,

Algorithm 4.1 Estimation of trial number in RANSAC

Require: Set of data

Input: Number of trials $k = \infty$

1: **while** $k >$ number of sample **do**

2: $w \leftarrow \dfrac{\text{number of good data points}}{\text{cardinality of dataset}}$

3: $k \leftarrow \dfrac{\log(1-z)}{\log(1-w^n)}$

4: number of sample $\leftarrow$ number of sample $+ 1$

Table 4.5: Set of data points for RANSAC example

Point	1	2	3	4	5	6	7
x	0	1	2	3	3	4	10
y	0	1.2	1.8	2.2	3.2	4	2.2

where $nX > n_m$ and n_m is the minimal sample set mentioned earlier in this section.

The following items explains the RANSAC algorithm rigorously:

1. Initiate the number of the sample (The minimum sample for a straight line is two), a threshold distance (dTh), iteration number, and best inlier number (zeros for the beginning). In addition, a minimum number of points to be considered as a consensus set is computed by

$$nCs = \text{round}(wnX). \tag{4.13}$$

2. Select n_m samples randomly. The red circles in Fig. 4.11.a are the chosen sample, namely $\boldsymbol{X}_a$ and $\boldsymbol{X}_b$.

3. Fit a line between two samples as shown in Fig. 4.11.b and compute its direction vector $\boldsymbol{qX}$

$$\boldsymbol{qX} = \frac{\boldsymbol{X}_b - \boldsymbol{X}_a}{\| \boldsymbol{X}_b - \boldsymbol{X}_a \|} \tag{4.14}$$

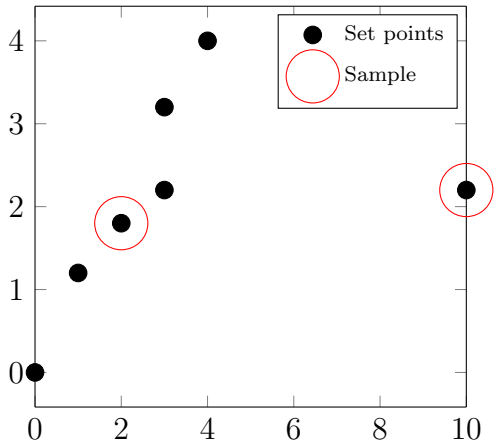
and its normal vector

$$\boldsymbol{qXn} = \left[-qX_y, qX_x\right]^{\mathrm{T}} \tag{4.15}$$

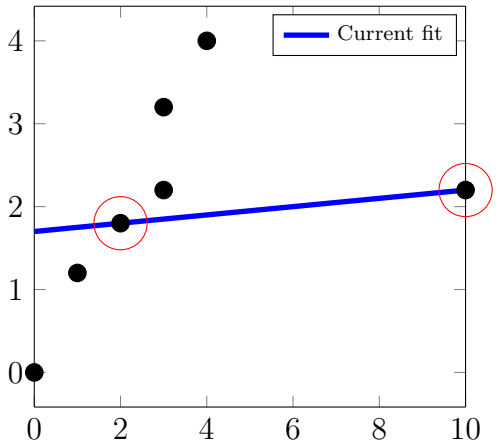
where qX_x and qX_y are the $x-$ and $y-$component of $\boldsymbol{qX}$.

4. Compute the error function by

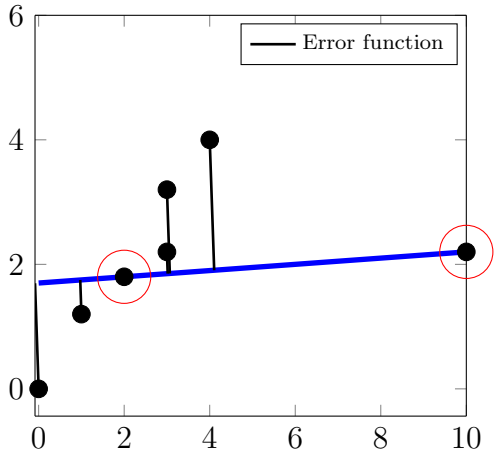
$$ef_i = \boldsymbol{qXn}^T (\boldsymbol{X}_i - \boldsymbol{X}_1) \tag{4.16}$$



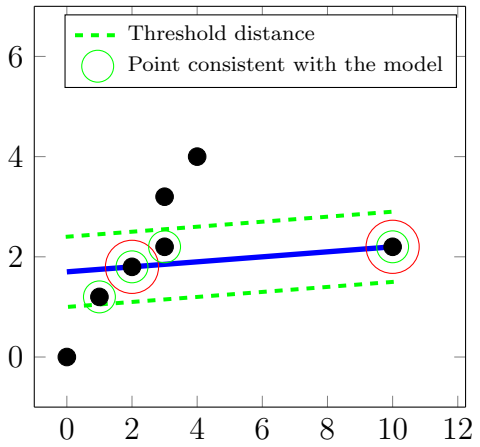
a. Sample Selection.



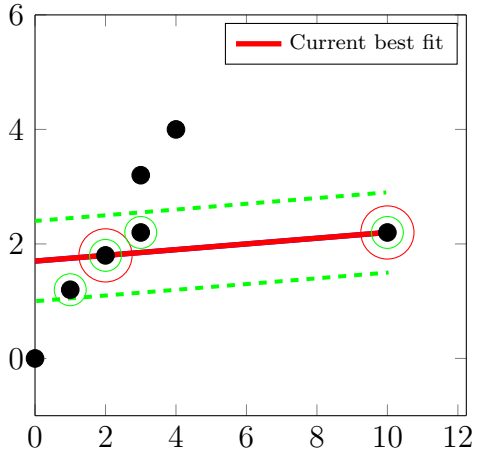
b. Line fitting.



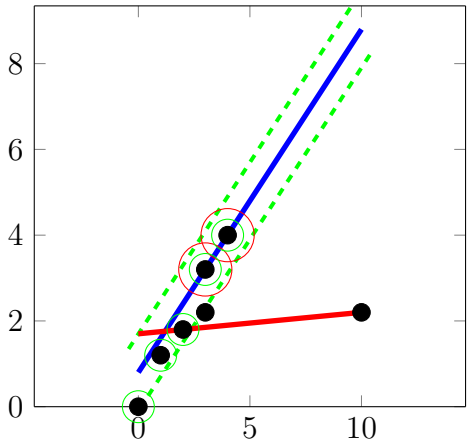
c. Error calculation.



d. Inlier selection



e. Comparing the inlier number



f. Iteration process

Figure 4.11: RANSAC line fitting step.

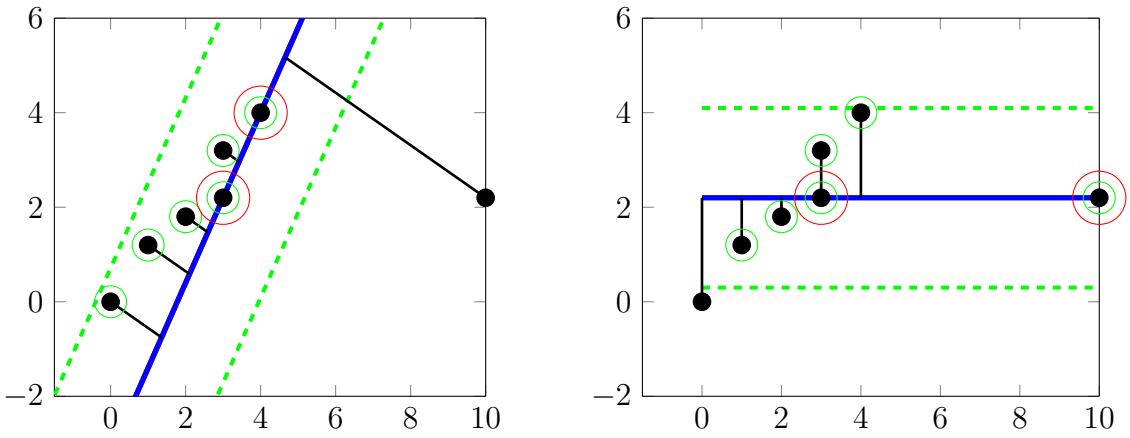


Figure 4.12: Too high distance threshold.

where $\{i \in \mathbb{Z} \mid 0 < i \leq nH\}$. The error function is the shortest distance from a data point to the fitted line. In Fig. 4.11.c, the shortest distance is the perpendicular distance from the data point to the fitted line (blue line).

5. Compare the error function with the threshold distance. If the error function is less than the threshold, then consider it as a temporary inlier, or formulated as the following

$$\boldsymbol{Xt} = \begin{cases} \boldsymbol{Xt}_i & \text{if} \quad ef_i \leq dTh \\ [\,] & \text{otherwise} \end{cases} . \tag{4.17}$$

Beside the mathematical description, an identical description is given in Fig. 4.11.d for easier human reading. Two green-dashed lines in parallel to the fitting line with distance dTh are drawn as boundaries. The points within the boundaries are defined as the current inliers.

6. Compare the cardinality of the current inliers with nCs and the best inlier number. The current inliers become the current consensus set if their cardinality exceeds both comparators. If this condition is satisfied, the model parameters[6] of the fitted line in the second step is denoted as the current best fit.

7. Repeat steps 2 to 6 for the new sample, as illustrated in Fig. 4.11.f, until the termination condition (for instance the iteration number) is achieved.

It is important to note the following about the selection of the dTh value. Too large a dTh leads to a condition where the boundary condition in step 6 above is satisfied for different sample choices but their model parameters are different, as illustrated in Fig. 4.12. In contrast, if dTh too small, then the boundary condition in step 6 is never fulfilled. Fig. 4.13 gives an illustration where the inliers are only a

[6]for a straight line, the model parameters are the slope and constant as in Eq. 3.32

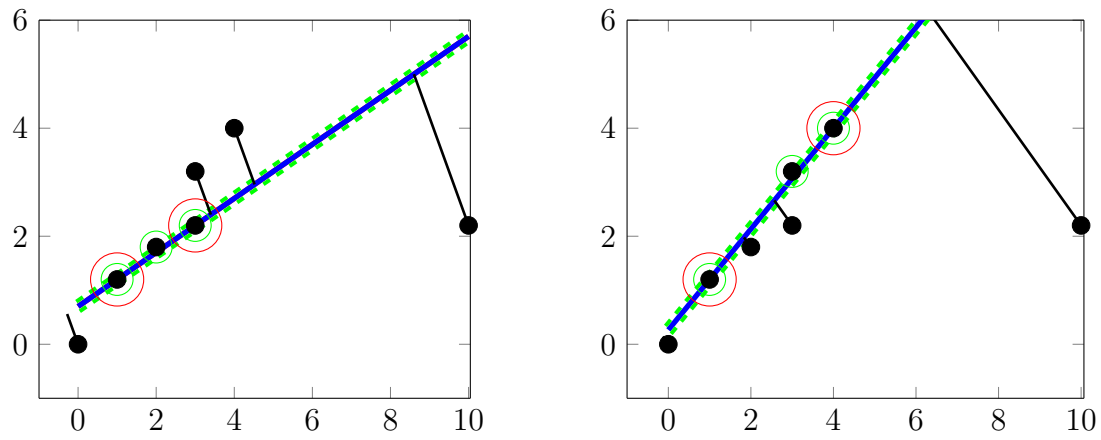


Figure 4.13: Too low distance threshold.

few points. If the boundary condition is not satistifed for all iterations, RANSAC generates no result.

4.3 Object segmentation

The laser scanner is used widely as a tool for environment perception. To extract the desired data from the measurement result, it is necessary to apply an object segmentation step. For instance, the laser scanner is mounted in the car as one of the measurement tools for object identification (car, truck, or pedestrian) or object tracking (car velocity, truck velocity). Three popular segmentation methods among researchers who conduct measurements using a 2D laser scanner are **I**terative **E**nd **P**oint **F**it (IEPF), **L**ine **T**racking (LT), and **S**uccesive **E**dge **F**ollowing (SEF) [Siadat et al., 1997]. In the following subsections, these methods are discussed.

4.3.1 Iterative End Point Fit

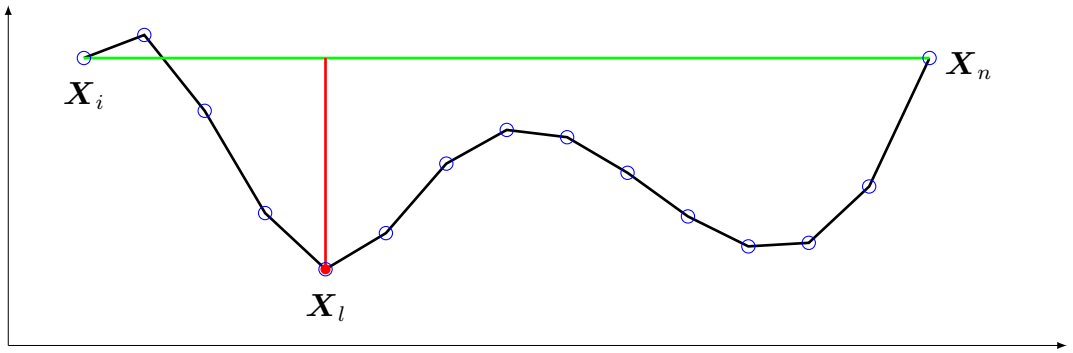
Iterative **E**nd **P**oint **F**it (IEPF) is also known as the *Ramer–Douglas–Peucker* algorithm and the *Split-and-Merge* algorithm. As its name indicates, this approach was proposed by several researchers. Ramer [1972] applied this approach to estimate the optimum number of vertices that lie on a plane curve. One year later, Douglas and Peucker [1973] published a similar idea to eliminate an unnecessary point along a straight line to minimize the use of computer memory. Although both ideas are identical, there is no reference in Douglas and Peucker [1973] to Ramer [1972]. A brief overview of the algorithm is given in the following.

Suppose the IEPF is performed to the set of data $\boldsymbol{X} = [x, y]^T \in \mathbb{R}^2$ as depicted in Fig. 4.14. The process is started by fitting the line between the first point $\boldsymbol{X}_i$ and the last point $\boldsymbol{X}_n$, and then followed by computing its perpendicular distance to all points[7]. If the maximum perpendicular distance exceeds a distance threshold, which is defined at the beginning, split the data into two sets. In this case, the perpendicular distance is a maximum at the point $\boldsymbol{X}_l$, which is illustrated as the red line in Fig. 4.14.a. Then, the data is divided into two sets: the first set is $\boldsymbol{X}_i$–$\boldsymbol{X}_l$ and the second set is $\boldsymbol{X}_l$–$\boldsymbol{X}_n$. Keep the first data set as a segment and use the second data set for further processing. Repeat the same process until the maximum distance is less than the threshold. The final result of this algorithm is depicted in Fig. 4.14.b, where four line segments are separate d by red point. In addition, a further process can be performed to fit these data segments more precisely, for instance, by the method of least squares.

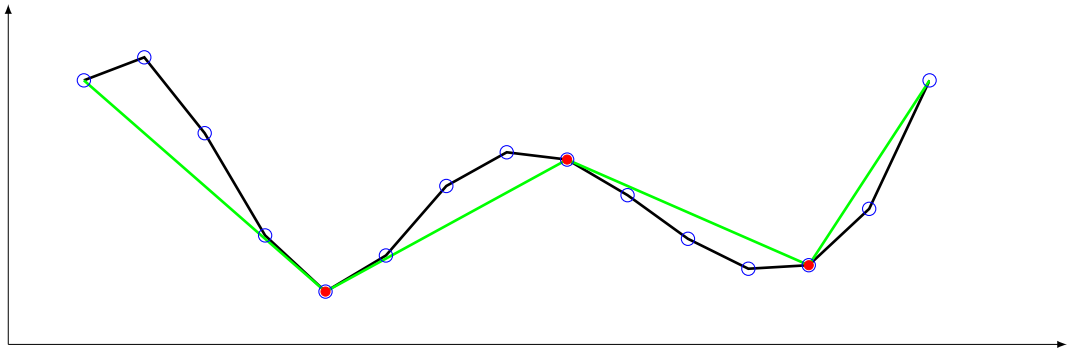
4.3.2 Line Tracking

Line **T**racking (LT) has a different name from Incremental Algorithm [Nguyen et al., 2005]. This approach is more or less similar to the IEPF, but the way to

[7] A similar calculation process can be found in steps 3 to 5 of the RANSAC algorithm in Section 4.2



a. Maximum perpendicular distance to the fitted line



b. Final line segment

Figure 4.14: IEPF algorithm.

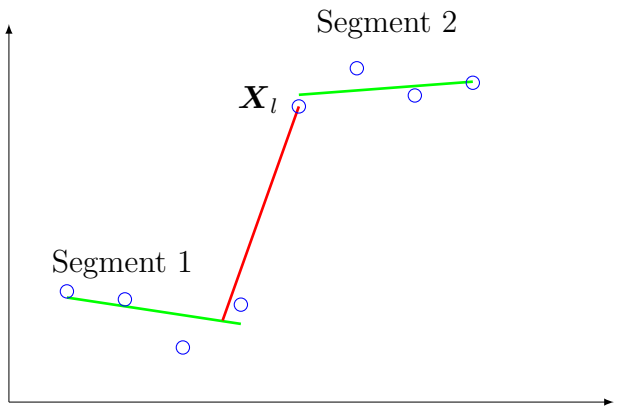


Figure 4.15: Line tracking algorithm.

choose the sample is different. The approach is described briefly in the following paragraph.

Consider the set of data in Fig. 4.15 where $\boldsymbol{X} = [x, y]^T \in \mathbb{R}^2$. At the beginning of the calculation process, a distance threshold is specified. Then, first, a linear regression is performed for the first and second point to obtain the model parameters of a line (the slope and intercept for the straight line). After that, the perpendicular distance from the fitted line with the next point is computed. If the distance is more than the threshold, split the points where the linear regression begins until it ends. Otherwise, restart the linear regression for new data which consists of the next point to the last point.

Figure 4.15 is a depiction of the LT. All the points before $\boldsymbol{X}_l$ are considered as a segment, because the perpendicular distance between its fitting curve and $\boldsymbol{X}_l$ (red line) exceeds the threshold. $\boldsymbol{X}_l$ becomes new starting point for a new linear regression.

4.3.3 Successive Edge Following

In contrast with IEPF and LT, **S**uccessive **E**dge **F**ollowing (SEF) is performed in a polar coordinate system. This is an advantage because the transformation from a Cartesian coordinate system is not necessary even if the data set are collected from a sensor in the Polar system, for instance, a laser range finder[8]. As an illustration, the data set is depicted in Fig. 4.16. Each point consists of $\boldsymbol{X} = [l, \alpha]^T$, where l and α are the distance of the point to the pole and the polar angle, respectively.

Similar to the other method above, a threshold is defined at the beginning. A segment is considered to be terminated if the absolute difference between the current

[8]Usually a laser range finder generates measurement result in polar coordinates

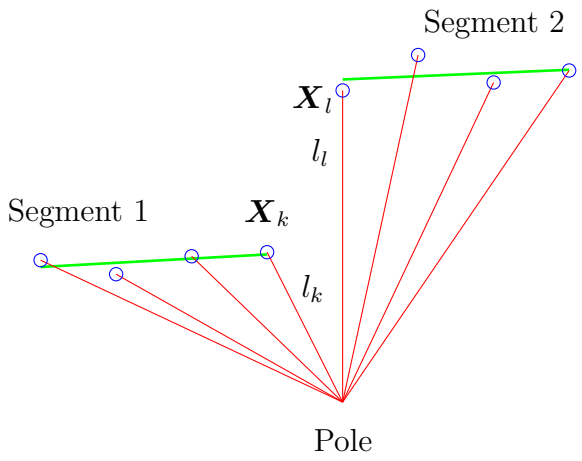


Figure 4.16: SEF algorithm.

with the next radius is more than the threshold, mathematically formulated by the following equation:

$$|l_k - l_l| > \text{Threshold}, \tag{4.18}$$

where the subscripts K and l represent the radius of the current and the posterior point. The current segment will be terminated at point $\boldsymbol{X}_k$ and the next process will be started from the next point $\boldsymbol{X}_l$. A similar process is repeated until the last point.

4.3.4 Proposed algorithm for object segmentation

By now, several algorithms have been discussed in the previous subsections. For the application in this research, none of those segmentation methods from subsection 4.3.1 to subsection 4.3.3 are robust to be applied for object segmentation without any modification to avoid a singularity. IEPF and LT has a singularity when all points have the same model parameters (gradient and intercept), as depicted in Fig. 4.5 (data sets within green rectangle). The termination condition will not be achieved even though the data set consists of four segments. Meanwhile, the SEF has a singularity when two point segments have mirror symmetry, as depicted in Fig. 4.17.

For the application in this work, a method to segment the measurement data is proposed. Consider the data set $\boldsymbol{X} \in \mathbb{R}^2$ depicted in Fig. 4.17 (without $\boldsymbol{X}_n$ and $\boldsymbol{q}_n$). A point, for instance $\boldsymbol{X}_k$, is considered to be a new segment if

$$\|\boldsymbol{X}_k - \boldsymbol{X}_{k-1}\| > c_1 Th_k \tag{4.19}$$

where c_1 is a constant and Th_k is the current threshold. The constant is defined at the beginning while the threshold must not be static because the length of the laser beam depends on the end-effector's position, in other words, the threshold has to

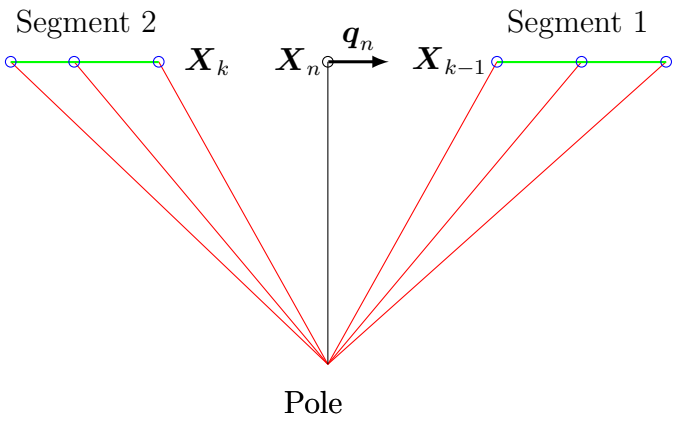


Figure 4.17: Proposed method for object segmentation.

be a function of the beam length. In [Siadat et al., 1997], the linear dependence of the distance between the current and the prior beam from the length of the laser beam and the anglular resolution is discussed. In the following paragraph, the computation of the threshold will be explained.

Suppose three or more successive laser beams face one or more flat surfaces. The measurement result will be a set of collinear points, as depicted in Fig. 4.17, with normal parametrizations $\boldsymbol{X}_n$ and direction vectors $\boldsymbol{q}_n$ known from the previous step[9]. The current threshold is formulated by

$$Th_k = \|\boldsymbol{X}_k - \widetilde{\boldsymbol{X}}_{k-1}\| \tag{4.20}$$

where $\widetilde{\boldsymbol{X}}_{k-1}$ is the computed prior point. The computed prior point is obtained from the intersection between $\boldsymbol{q}_n$ and the prior direction vector of the laser beam ($\boldsymbol{q}_{k-1}$), which is determined by rotating $\boldsymbol{X}_k$ about the selected angular resolution of the laser scanner $\Delta\Theta$ in the direction clockwise, as follows,

$$\boldsymbol{q}_{k-1} = \boldsymbol{R}\boldsymbol{X}_k, \tag{4.21}$$

where $\boldsymbol{R}$ is the 2D rotation matrix for the clockwise direction given in Eq. 3.48. The computation of the intersection point between two vector is explained in Appendix A.

[9]In this research, the line parameters are obtained from the previous step. The normal parametrization and line parameters are extracted from the collinear points by the Hough Transform and linear least squares.

4.4 Application of the line extraction algorithm for CABLAR

To design the strategy, including selecting the suitable method of object segmentation, the typical measurement data must be known. Consider a set of lengths of beams l_i, where $\{i \in \mathbb{Z} \mid 0 < i \leq n_b\}$, is measurement data from laser scanner LMS500 in polar coordinates. The angle θ_i between l_i and the horizontal axis (the x-axis in the Cartesian coordinate system or 0° in the polar coordinate system) is calculated by Eq. 3.15. In the Cartesian coordinate system, (l_i, θ_i) is represented by $\boldsymbol{h}_i \in \mathbb{R}^2$ and depicted in Fig. 4.18. The laser scanner is mounted on the end-effector

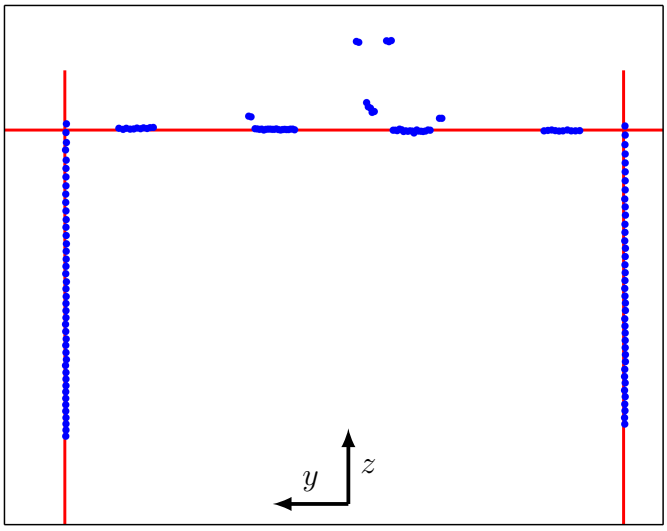


Figure 4.18: Measurement data.

and its beam heading to the reflector with special pattern design as depicted in Fig. 3.4. The environment inside the working space and its boundary (reflectors) is static. There is no foreign object to suddenly come into the working space.

The strategy to extract the normal parametrization of the measurement data belonging to the reflectors is described in the following:

1. The normal parametrization of the left- (d_{hl}, θ_l), right- (d_{hr}, θ_r), and upper-side (d_{hu}, θ_u) reflectors is extracted using the Hough Transform. Then, three red straight lines are visualized in Fig. 4.18 according to the normal parametrization. In the Cartesian coordinate system, these normal parametrizations of the left, right, and upper side are written as $\boldsymbol{X}_{hl}$, $\boldsymbol{X}_{hr}$ and $\boldsymbol{X}_{hu}$, respectively. Fig. 4.19 is an enlargement of Fig. 4.18. Here, the drawback of the Hough Transform due to the grid size appears. The gradient of the red line (from the HT) is not the best fit of the collinear points (measurement data).

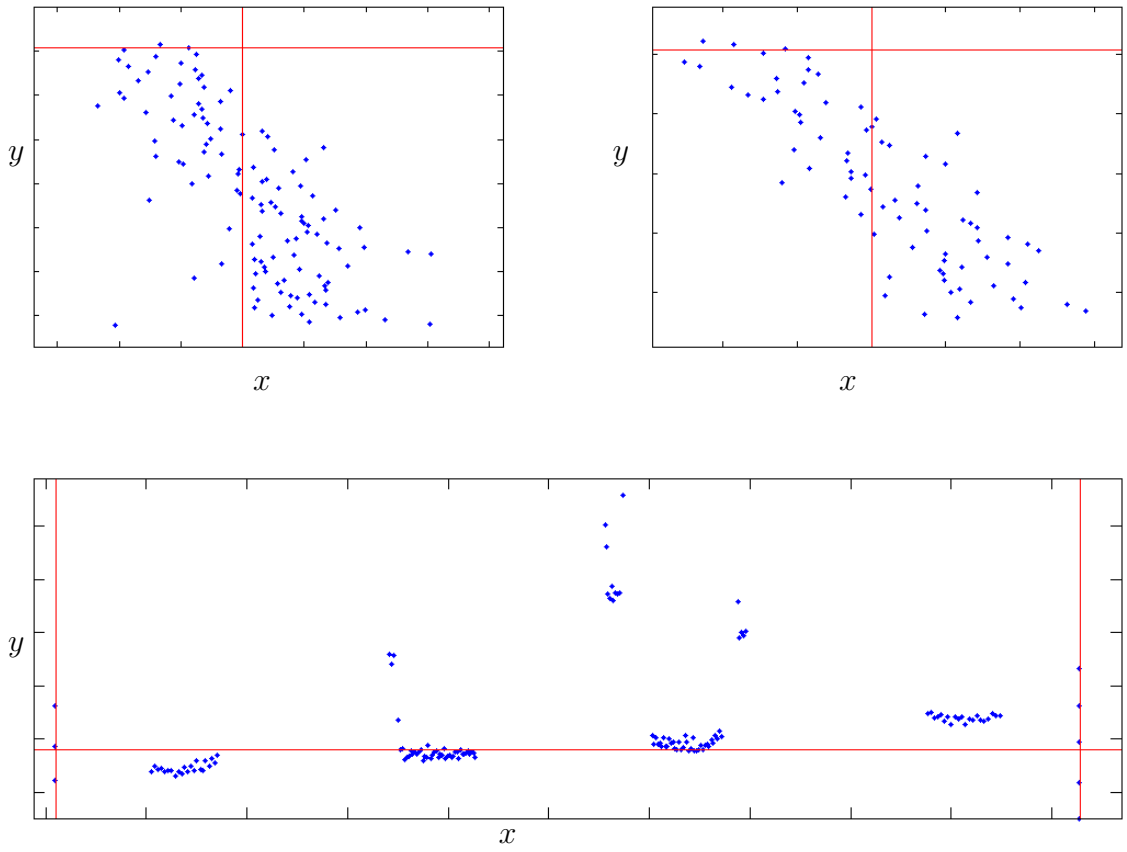


Figure 4.19: Zoom Measurement result.

2. The measurement points are segmented according to the reflector position by means of which point belongs to which reflector. This step proceeds by calculating two corner points. The first is the intersection between the straight lines of the right-side reflector with the straight lines of the upper-side reflector, denoted by $\boldsymbol{X}_{cr}$. The second is at the intersection between the straight lines of the left-side reflector and the straight lines of the upper-side reflector, denoted by $\boldsymbol{X}_{cl}$. Appendix A describes how to calculate the intersection of two vectors. The position vectors for this case are the normal parametrization in Cartesian coordinates X_{hl}, X_{hr}, X_{hu} and its direction vector is

$$\boldsymbol{n} = [-y, x]^T \tag{4.22}$$

where x and y are the components of its normal parametrization in Cartesian coordinates.

The corner points $\boldsymbol{X}_{cr}$ and $\boldsymbol{X}_{cl}$ are now converted into polar coordinates and denoted by (d_{cr}, θ_{cr}) and (d_{cl}, θ_{cl}), where the subscripts l and r stand for left and right, respectively. For the sake of efficiency in the calculation (to save computer memory), only the indices are stored instead of creating a new matrix for the further step. Since the angles θ_{cr} and θ_{cl} are known, the indices are calculated by Eq. 3.15 and the notation i_{cr} and i_{cl} is used for the indices. Please note that rounding is needed to obtain an integer result. Finally, the indices associated with a specific reflector are obtained as follows.

$$\begin{array}{lcll} \mathcal{I}_R & = & \{1, 2, 3, ..., i_{cr}\} & \text{for the right side} \\ \mathcal{I}_U & = & \{i_{cr}+1, i_{cr}+2, i_{cr}+3, ..., i_{cl}-1\} & \text{for the upper side} \\ \mathcal{I}_L & = & \{i_{cl}, i_{cl}+1, i_{cl}+2, ..., n_M\} & \text{for the left side} \end{array} \tag{4.23}$$

3. The measurement points specified by the indices in the point above still have outliers. In this step, the outliers are going to be eliminated from the data set using RANSAC. The normal parametrization and the corner point are chosen instead of a random point as the sample in RANSAC, for instance, the normal parametrization of the right-side reflector $\boldsymbol{X}_{hr}$ and the corner point $\boldsymbol{X}_{cr}$ are chosen as the sample when RANSAC is applied to the points belong the right reflector . Repetition with a different sample is not necessary here because the aim of the RANSAC application here is not to fit the line, but to remove the outliers. The outlier-free indices are denoted by $\widetilde{\mathcal{I}_R}$, $\widetilde{\mathcal{I}_U}$, and $\widetilde{\mathcal{I}_L}$.

4. The linear least squares method is applied to the measurement data specified by $\widetilde{\mathcal{I}_R}$, $\widetilde{\mathcal{I}_U}$, and $\widetilde{\mathcal{I}_L}$ to obtain the straight lines parameters. Then, the normal and direction vectors are computed from the straight line parameters. Appendix B describes how to calculate these vectors. The new normal parametrization for the left-, right-, and upper-side reflectors are denoted by

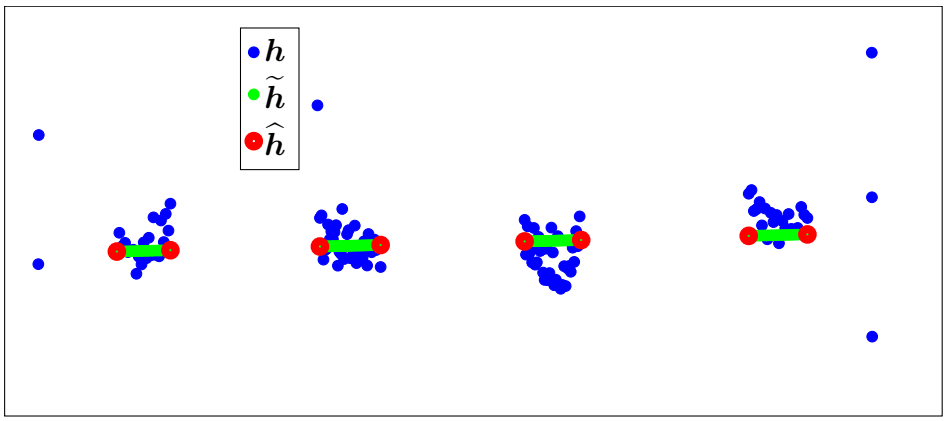


Figure 4.20: Pre and post processing of measurement data.

$\widetilde{\boldsymbol{X}}_{hl}$, $\widetilde{\boldsymbol{X}}_{hr}$ $\widetilde{\boldsymbol{X}}_{hu}$ in Cartesian coordinates and by $(\widetilde{d}_{hl}, \widetilde{\theta}_{hl})$, $(\widetilde{d}_{hr}, \widetilde{\theta}_{hr})$, $(\widetilde{d}_{hu}, \widetilde{\theta}_{hu})$ in polar coordinates.

The other important task in this step is the curve fitting of the measurement data indexed by $\widetilde{\mathcal{I}_U}$ in order to reduce the measurement noise. The curve is fitted based on the straight line parameters from the least squares above and denoted by $\widetilde{\boldsymbol{h}}$, shown in Fig. 4.20.

5. The new corner points are computed from the new normal parametrization $\widetilde{\boldsymbol{X}}_{hl}$, $\widetilde{\boldsymbol{X}}_{hr}$ and $\widetilde{\boldsymbol{X}}_{hu}$, denoted by $\widetilde{\boldsymbol{X}}_{cr}$ and $\widetilde{\boldsymbol{X}}_{cl}$. Refer to point 2 above to compute the new intersection point.

6. The algorithm in subsection 4.3.4 is applied to the data set $\widetilde{\boldsymbol{h}}$ with the purpose of segmenting the data set. Since the reflector at the upper-side consists of four segments n_s, the data set are also segmented into the same number. Only the first and last points of each segment are considered for the further process in Section 3.5 and the rest are neglected. Fig. 4.20 illustrates the selected points, denoted by $\widehat{\boldsymbol{h}}_i \in \mathbb{R}^2$, where $\{i \in \mathbb{Z} \mid 0 < i \leq 2n_s\}$.

4.5 Platform pose and measurement delay

4.5.1 Platform position calculation

The position of the platform ${}^{B}\boldsymbol{r}$ is computed by rearranging Eq. 3.7 as

$$ {}^{B}\boldsymbol{r} = {}^{B}\boldsymbol{r}_{BL} - {}^{B}\boldsymbol{r}_{PL}. \tag{4.24}$$

${}^{B}\boldsymbol{r}_{PL}$ is computed by Eq. 3.8. The platform orientation yaw and roll angles are determined by the proposed method while the pitch angle is measured by the IMU. ${}^{B}\boldsymbol{r}_{BL}$ is extracted from the measurement data. In detail, the x-component

of ${}^{B}\boldsymbol{r}_{BL}$ is calculated by the mathematical model in Section 3.5 while the y- and z-components are computed by

$$\begin{aligned} r_y &= \frac{\widetilde{d}_{hr} - \widetilde{d}_{hl}}{2} \cos\psi \\ r_z &= \left(\Delta z - \widetilde{d}_{hu}\right) \cos\vartheta. \end{aligned} \tag{4.25}$$

4.5.2 Measurement delay compensation

The determined platform position in subsection 4.5.1 is transmitted to a module in the automation software TwinCAT 3 through TwinCAT ADS. The transmission process is mentioned in subsection 2.3.2 and Fig. 2.14. Due to time delay, the module receives a not actual determined pose. The scanning process in a laser scanner, transferring the measurement data from the laser scanner to the host PC, and processing the measurement data to extract the platform pose, require a certain amount of time. Consequently, a compensation must be proposed to determine the actual platform position, which is explained in the following paragraph.

Suppose the determined platform position ${}^{B}\boldsymbol{r}_{k\text{-}1}$ is received by the module at time t_k[10]. t_k is obtained from the TwinCAT 3 internal time. Then, the actual platform position is determined by

$${}^{B}\boldsymbol{r}_{k} = {}^{B}\boldsymbol{r}_{k\text{-}1} + \Delta\boldsymbol{r}, \tag{4.26}$$

where $\Delta\boldsymbol{r}$ is the vector of the platform motion during the time delay Δt and is computed by

$$\Delta\boldsymbol{r} = \frac{\boldsymbol{v}_k + \boldsymbol{v}_{k-1}}{2} \Delta t. \tag{4.27}$$

$\boldsymbol{v}_k$ and $\boldsymbol{v}_{k-1}$ are the desired actual and prior translational platform velocities, respectively. The time delay Δt is obtained from

$$\Delta t = t_k - t_{k-1}, \tag{4.28}$$

where t_{k-1} is the time when the laser scanner starts the measurement and generated by the laser scanner as a time stamp.

4.6 Concluding remarks

Hough Transform and Random Sample Consensus has been combined to select the desired data from the measurement data transmitted by the laser scanner. The

[10]Subscripts k and $k-1$ indicate the actual and prior state, respectively.

measurement data is considered as the desired data if the measurement data corresponds to the reflector at the left, right and upper side of robot frame. Then, the desired data is fitted by the least square method to obtain the normal parametrization which used later to compute two translational components and one rotational component of the platform pose. The data corresponded with the upper reflector is segmented by the proposed object segmentation method. The rest translational component and one rotational component of the platform pose is extracted from the segmented data. An algorithm to compensate the measurement delay due to certain process before the determined pose is received by the module in the robot control algorithm has been developed.

CHAPTER 5

Simulation and Experimental Results

In this chapter, the simulation and experimental results will be presented and discussed. The desired values, such as the desired platform trajectory, and the measured values, such as the simulation and measurement results, are going to be compared in order to give an idea of the accuracy of the algorithm and the robot. Instead of displaying the desired and measured values on the same graph, the comparison of the desired and measured values is presented in terms of the absolute error

$$\text{absolute error} = |\text{desired value} - \text{measured value}| \tag{5.1}$$

to give a better overview for the reader.

According to Lalo [2013], the workspace of CABLAR is depicted in Fig. 5.1. The blue circles are the winches. The feasible robot workspace is within by the blue rectangle. The feasible workspace is determined by the wire force, which must be within the defined range. The platform motion for the simulation and experiment in the following sections will be defined within this feasible workspace.

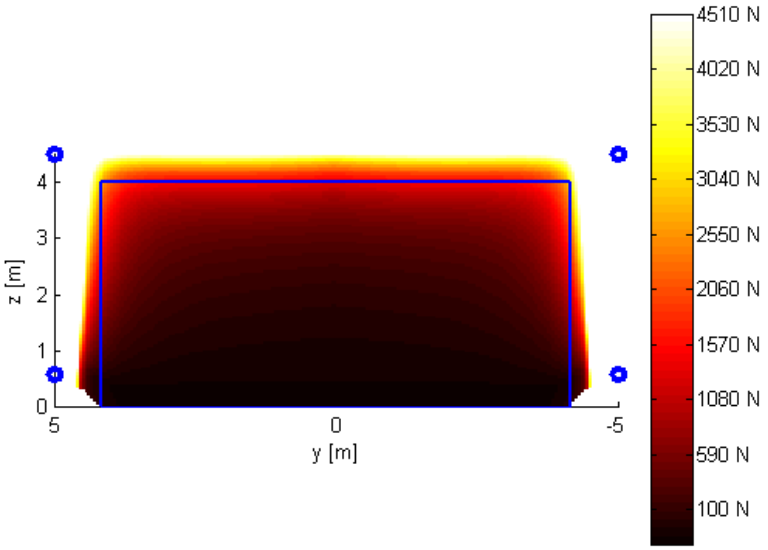


Figure 5.1: Workspace of the robot [Lalo, 2013].

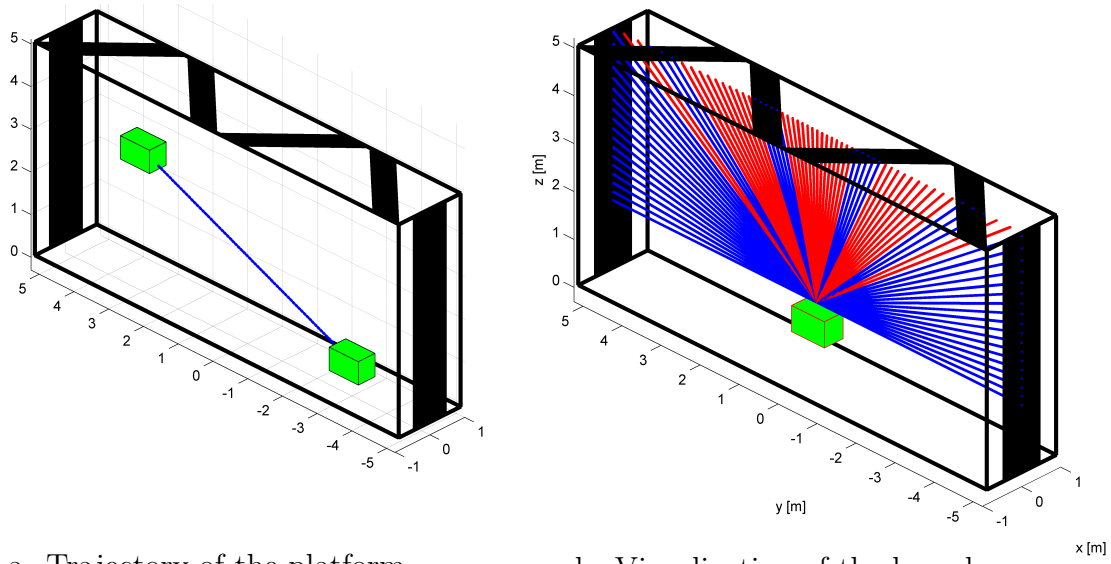
5.1 Simulation results

The simulation was performed before the experiment to obtain the algorithm's accuracy when the platform is stationary and during the motion. Several parameters, such as the platform velocity and the diameter of the laser beam, are going to be tested in order to observe their influence on the 5 aspects of the platform pose, the x-component, y-component, z-component, roll angle, and yaw angle.

An example of a typical platform motion is depicted in Fig. 5.2.a, where the platform moves diagonally from $\boldsymbol{x}_a = [0, -4, 0.5]^{\mathrm{T}}$ to $\boldsymbol{x}_b = [0, 4, 3]^{\mathrm{T}}$. In addition, the simulation was also performed for the whole robot workspace. Since the laser beam is invisible in a real system, a visualization from the simulation is depicted in Fig. 5.2.b. to give an idea of how the laser scanner works. The intersection of the beams with the desired reflector are presented in blue while the rest are red. The components of the platform orientation (yaw, pitch and roll angle) are defined to be constant at 0° for any simulation in this section.

The velocity effect is not taken into account when the platform is stationary. For that reason, the position of the laser scanner's origin with respect to the robot inertial system ${}^{B}\boldsymbol{r}_{BL}$ is similar for all laser beams for each scan, as depicted in Fig. 5.3.a. In contrast, ${}^{B}\boldsymbol{r}_{BL}$ is not the same for all laser beams during the platform's motion when the velocity effect is taken into account. The origin of each laser beam is computed by Eq. 3.21 and is depicted in Fig. 5.3.b.

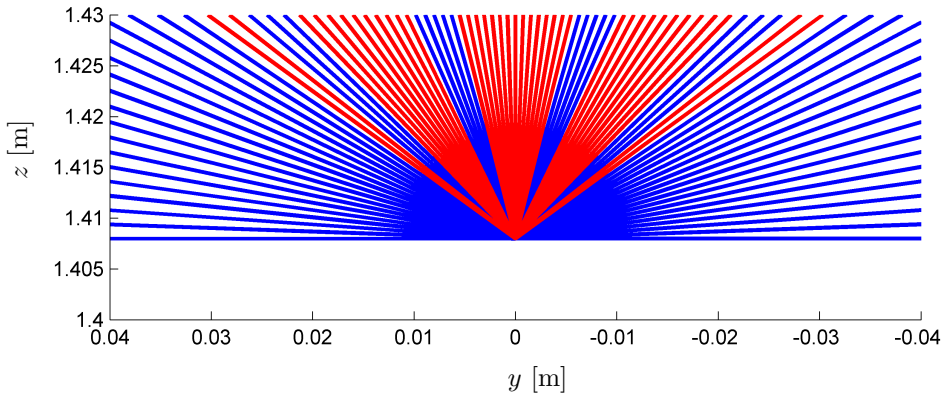
The effect of the velocity on the laser beam's origin with respect to the robot inertial system can be seen clearly in Fig. 5.3, where the origin of the laser beams are not the same. Consequently, the measurement result from the laser is also



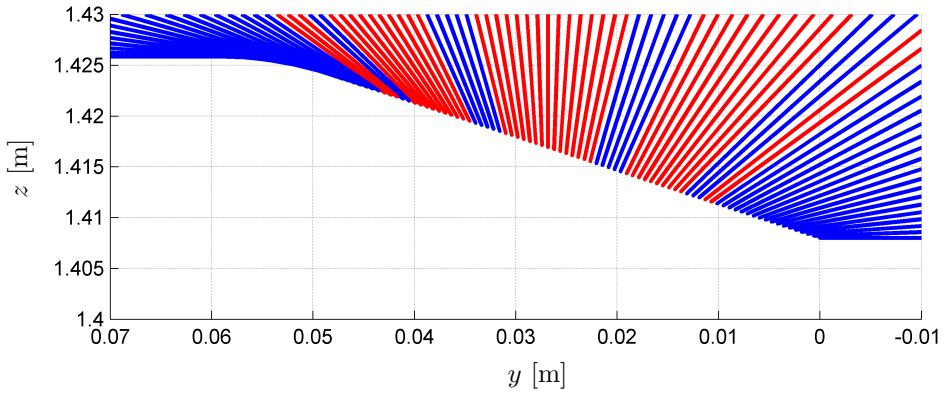
a. Trajectory of the platform

b. Visualization of the laser beams

Figure 5.2: Trajectory of the platform.



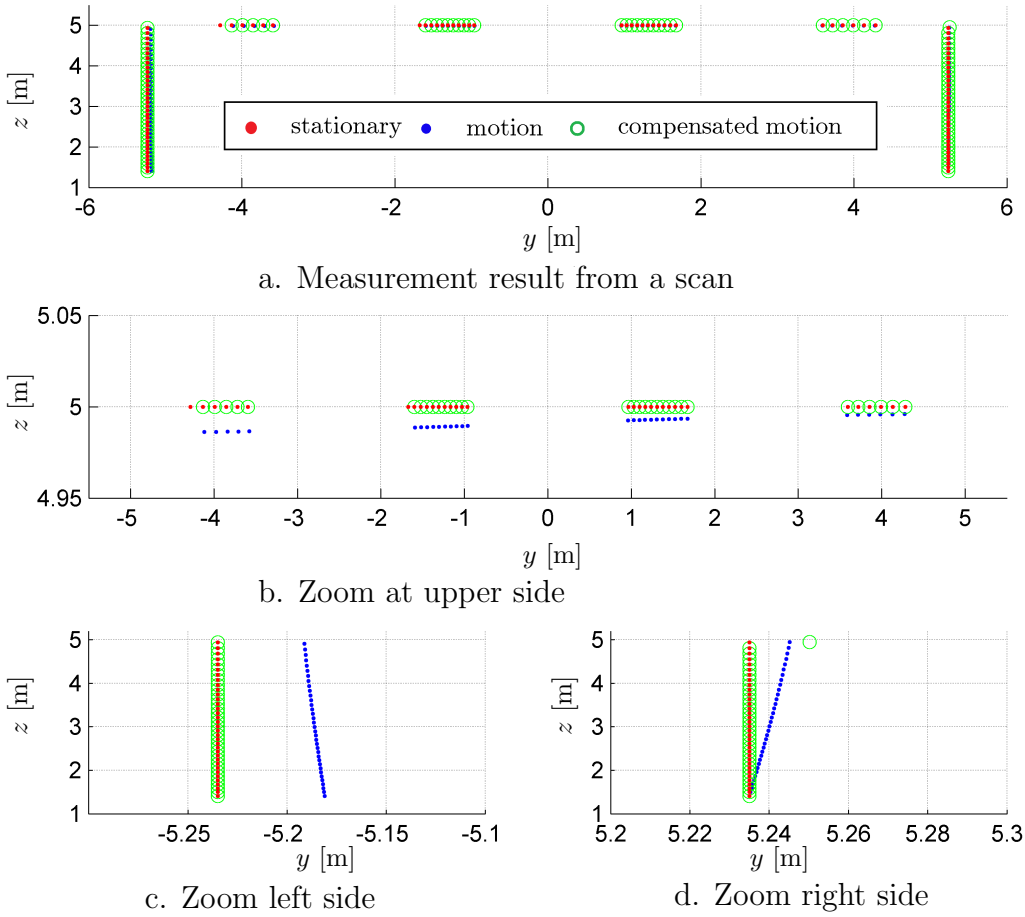
a. Without velocity effect



b. With velocity effect

Figure 5.3: Visualization of origin of laser scanner with respect to robot inertial system.

influenced by the velocity effect. A comparison of the measurement data of a scan with (denoted as motion) and without (denoted as stationary) the velocity effect is depicted in Fig. 5.4. In the simulation with the velocity effect, the platform



a. Measurement result from a scan

b. Zoom at upper side

c. Zoom left side

d. Zoom right side

Figure 5.4: Comparison of static and dynamic measurement result.

is simulated with a constant speed of 4 $\mathrm{m.s}^{-1}$. To compensate for the velocity effect, the mathematical model in Section 3.2 is applied and it outputs the result as depicted in Fig. 5.4. A comparison of those results shows that the mathematical model in Section 3.2 works properly to reduce the measurement deviation due to the platform's motion for the vertical and horizontal measurement points. However, the number of laser beams which intersect with the reflector at the upper side between the simulation results in the stationary condition and after compensation are not the same. Consequently, this influences the estimated x-component of the platform position vector because the mathematical model requires the distance from the corner to the first and the last point of a measurement set from each reflector at the upper side, as mentioned in Section 3.5 and depicted in Fig. 3.8.

Equally important for the result of estimating the x-component is the diameter of the laser beam. In a real system, the diameter of the laser beams depends on the

beam's length and influences the measurement result. To observe the influence of the beam's diameter on the estimated x-component, a simulation with a constant beam diameter and with a beam diameter dependent length has been performed.

Moreover, to distinguish the estimated results of all translational components of the platform position vector and to avoid an overcrowded figure, simulations with and without the velocity effect will be explained separately in the following subsections. If the figure shows more than one graph with a similar horizontal axis, its horizontal label is written only on the upper graph. The other plot is to be understood as having the same label. For instance, the label of the upper horizontal axis of all plots in Fig. 5.5 is written only in Fig. 5.5.a, while the label of the lower horizontal axis is written only in Fig. 5.5.c.

5.1.1 Without velocity effect

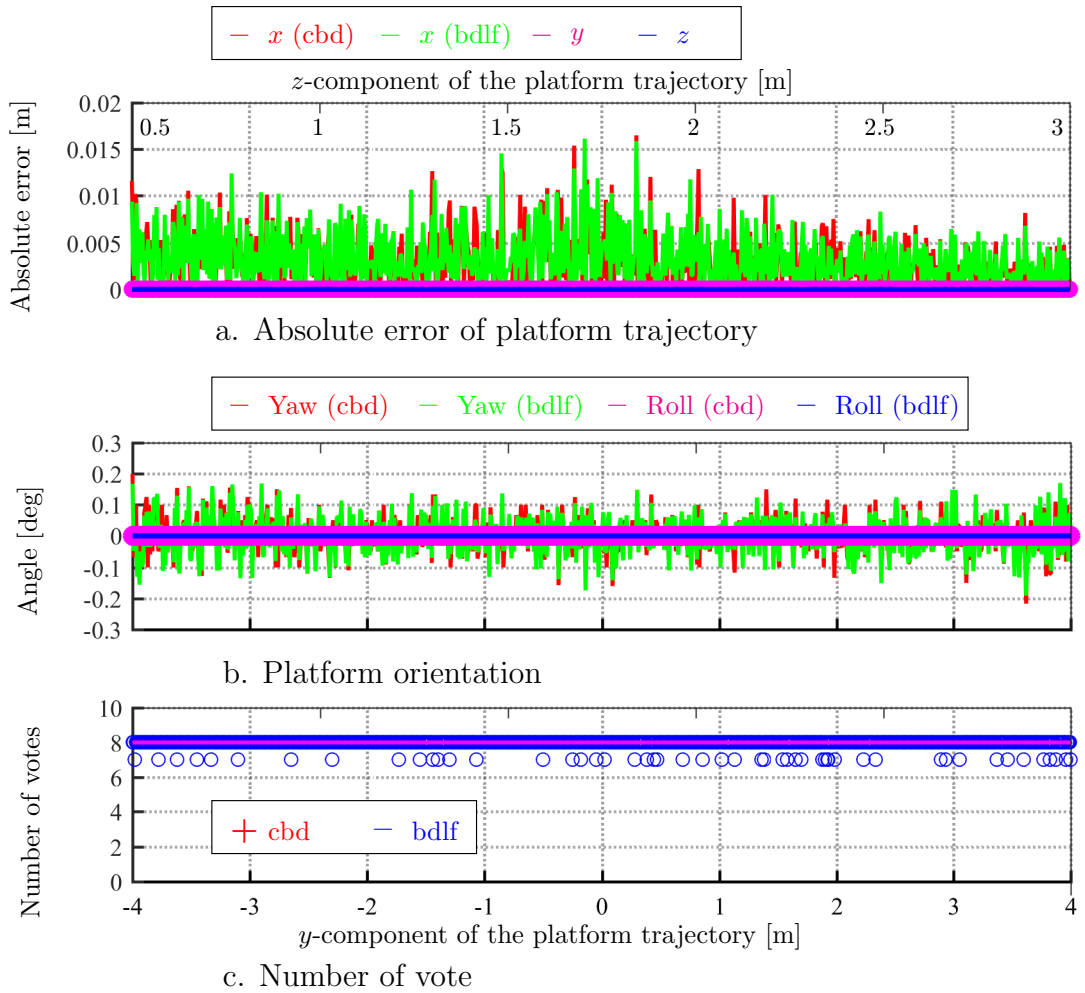
The simulation results of the platform trajectory as depicted in Fig. 5.2 are shown in Fig. 5.5. The lower and upper horizontal axes of the graph in Fig. 5.5 are the y- and z-components of the platform trajectory.

Fig. 5.5.a shows that the absolute error of the y- and z-components are zero for any platform position. The absolute errors of the x-component for the simulations with and without the beam diameter effect are quite similar, with a maximum absolute error of 13 mm. The proposed algorithm also determines the yaw and roll angles of the platform orientation, which is shown in Fig. 5.5.b. The estimated roll angles are constant regardless of the beam diameter, while the estimated yaw angles fluctuate, but by less than $\pm 0.2\,^{\circ}$. Fig. 5.5.c compares the simulations with a constant beam diameter and with a beam diameter length function. The simulation result of the beam diameter length function has always a maximum number of votes, while the simulation without the beam diameter effect has a fluctuation in the number of votes, but still within an acceptable number.

A simulation for the whole robot workspace was performed in order to determine the feasible area of the algorithm within the robot workspace according to the absolute error of some components of the platform pose. The platform trajectory is discretized at 2 cm in the y- and z-axes, while x is constant at 0. The results are shown in Figs 5.6–5.10. The absolute error of the y- and z-components is depicted in Fig. 5.6.a and Fig. 5.6.b, respectively. These absolute errors are almost equal to zero. That means that the proposed method works perfectly to determine the y- and z-components of the translational position of the platform.

The proposed algorithm to estimate the y- and z-components also outputs the roll angle of the platform, where its absolute error within the robot workspace is shown in Fig. 5.7.

In summary, the results achieved show that the proposed method to estimate the roll (φ), the y-, and the z-component of the platform pose while the platform is



a. Absolute error of platform trajectory

b. Platform orientation

c. Number of vote

Figure 5.5: Absolute error of the translational components of the platform pose in stationary condition and the number of votes.

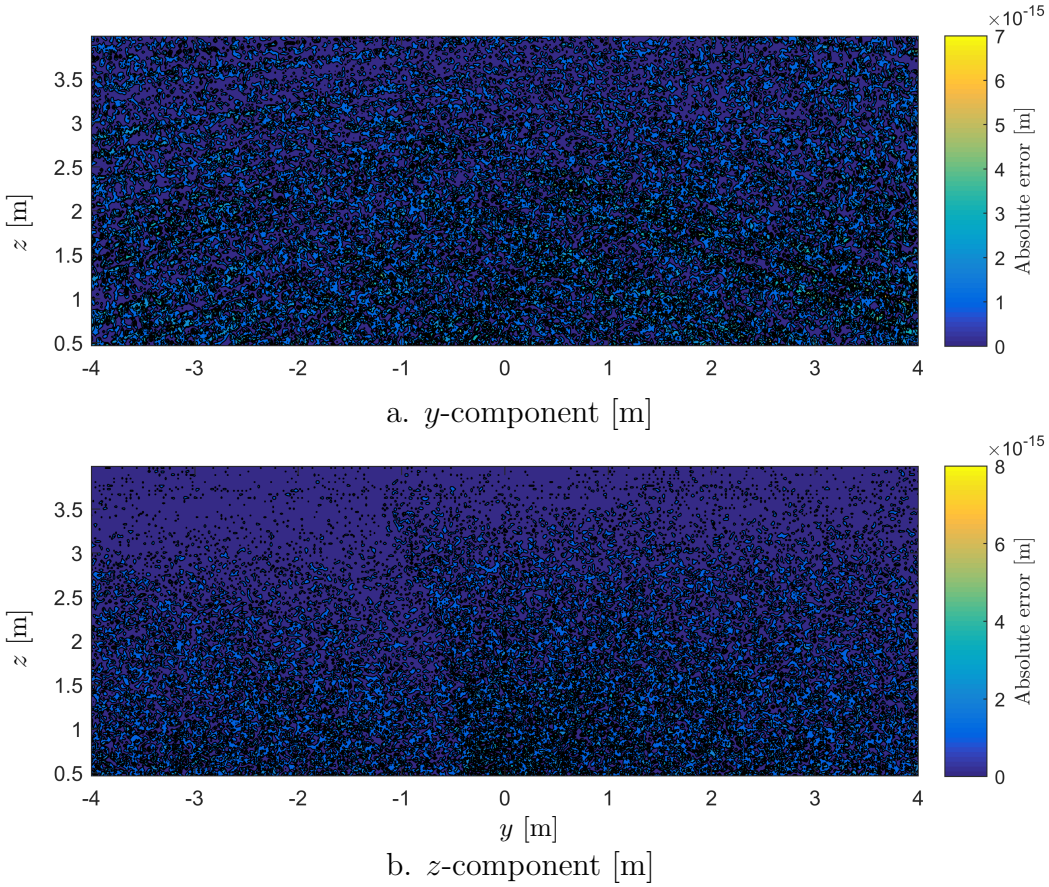


Figure 5.6: Absolute error of the y- and z-component in workspace.

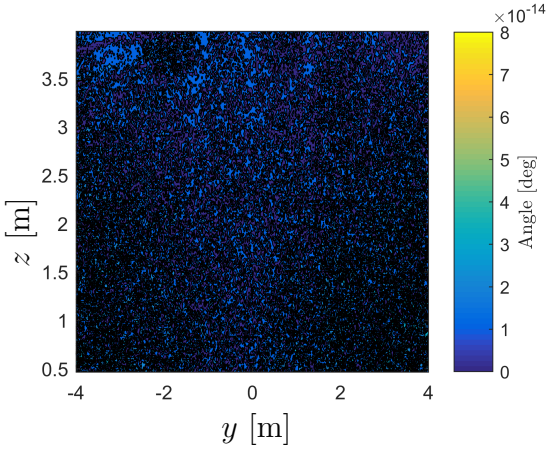
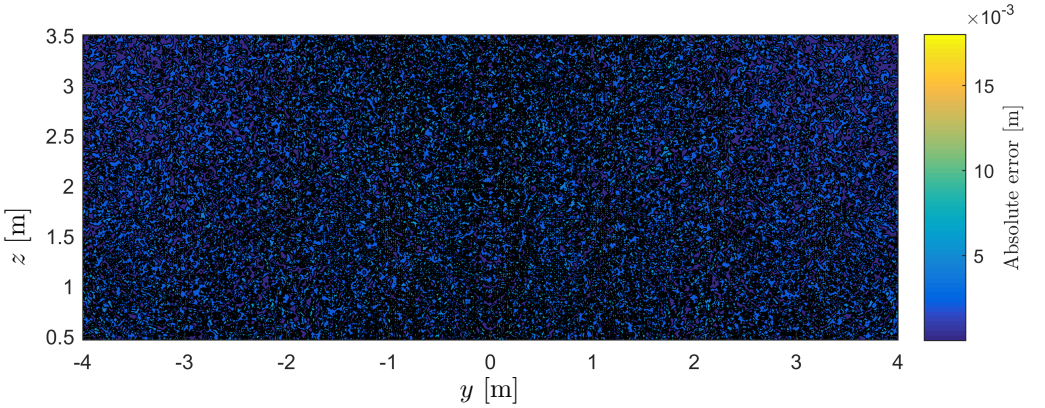


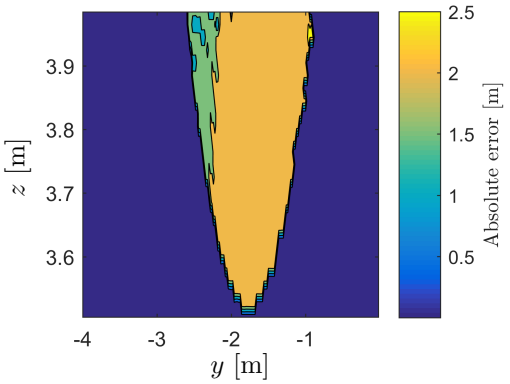
Figure 5.7: Roll angle.

stationary, works satisfactorily. It is an advantage for the next steps because the proposed algorithm to estimate the x-component requires the estimated y-, and z-components in the calculation, as mentioned in Section 3.5.

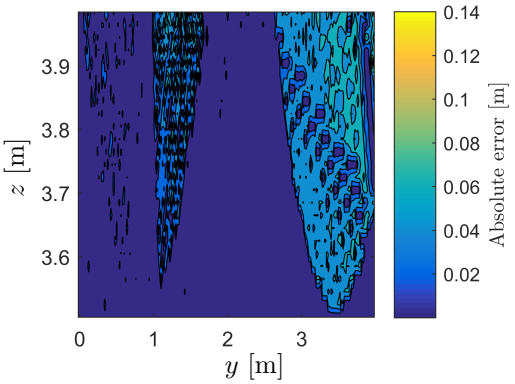
The absolute error of the x-component within the workspace $0.5\,\mathrm{m} \leq z \leq 3.5\,\mathrm{m}$ and $3.5\,\mathrm{m} < z \leq 4\,\mathrm{m}$ is depicted in Fig. 5.8.a and Fig. 5.8.b, respectively. In order



a. Workspace $0.5\,\mathrm{m} \leq z \leq 3.5\,\mathrm{m}$



b. Workspace $3.5\,\mathrm{m} \leq z \leq 4\,\mathrm{m}$
$-4\,\mathrm{m} \leq y \leq 0\,\mathrm{m}$

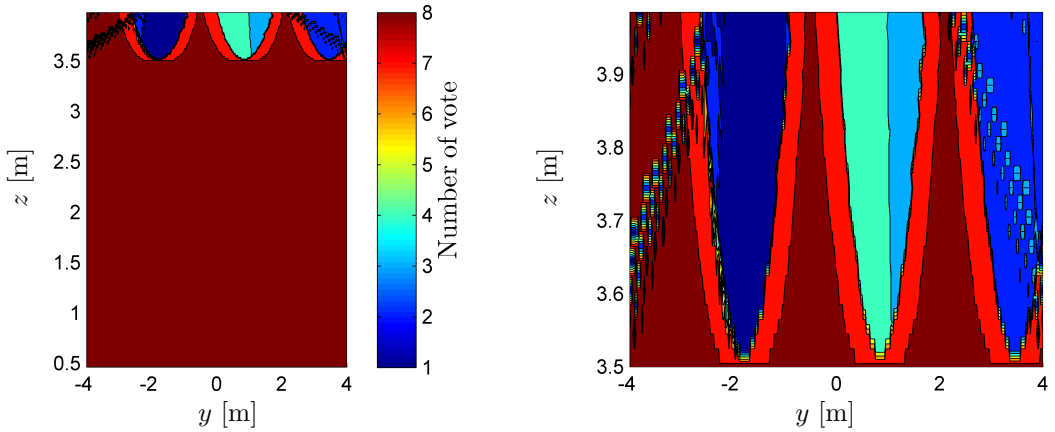


c. Workspace $3.5\,\mathrm{m} < z \leq 4\,\mathrm{m}$
$0\,\mathrm{m} \leq y \leq 4\,\mathrm{m}$

Figure 5.8: Absolute error of the x-component within the robot workspace .

to display the legend of the contour plot in two different scales, the result is shown in separate contour plots according to the z-component of the platform position. This allows the reader to see the error within the workspace clearly.

According to Fig. 5.8, the maximum absolute errors of the x-component for $0.5\,\mathrm{m} \leq z \leq 3.5\,\mathrm{m}$ and $3.5\,\mathrm{m} < z \leq 4\,\mathrm{m}$ are 15 mm and 2.5 m respectively. The error is coming from the measurement deviation due to angular resolution of the laser beam. The measured length of the reflectors at the upper side does not represent the actual length due to the measurement deviation as mentioned in Sec. 3.4. This absolute error corresponds with the number of votes as shown in Fig. 5.9. The number of vote is always maximum for $0.5\,\mathrm{m} \leq z \leq 3.5\,\mathrm{m}$ where the maximum absolute error of x-component is 15 mm. In contrast, the number of vote for



a. Number of vote on workspace b. Workspace $3.5\,\mathrm{m} < z \leq 4\,\mathrm{m}$

Figure 5.9: Number of votes within the robot workspace.

$3.5\,\mathrm{m} < z \leq 4\,\mathrm{m}$ fluctuates from 1 to 8 and the maximum absolute error of x-component of 2.5 m which considered wrong. Considering the absolute error, the proposed method is applicable for the region $0.5\,\mathrm{m} \leq z \leq 3.5\,\mathrm{m}$ and $-4\,\mathrm{m} \leq y \leq 4\,\mathrm{m}$.

The other pose component which corresponds with the number of votes is the yaw angle (ψ), shown in Fig. 5.10. The maximum absolute error of the yaw angle for $0.5\,\mathrm{m} \leq z \leq 3.5\,\mathrm{m}$ and $3.5\,\mathrm{m} < z \leq 4\,\mathrm{m}$ are $\pm 0.25°$ and $-0.25\,° \leq \psi \leq 50\,°$, respectively.

By now, all the desired platform poses have been determined[1]. Based on the result, the feasible area of the platform pose estimation within the robot workspace can be defined as follows:

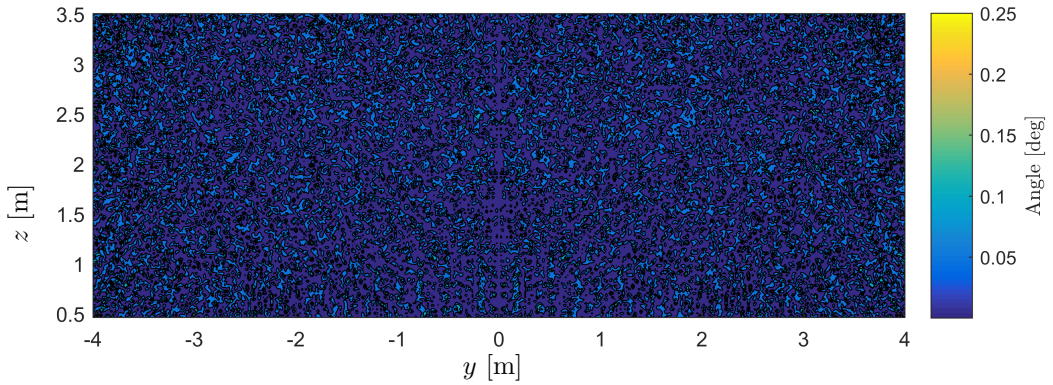
$$\begin{aligned} -4\,\mathrm{m} \leq y \leq 4\,\mathrm{m} \\ 0.5\,\mathrm{m} \leq z \leq 3.5\,\mathrm{m} \end{aligned} \quad . \tag{5.2}$$

5.1.2 With velocity effect

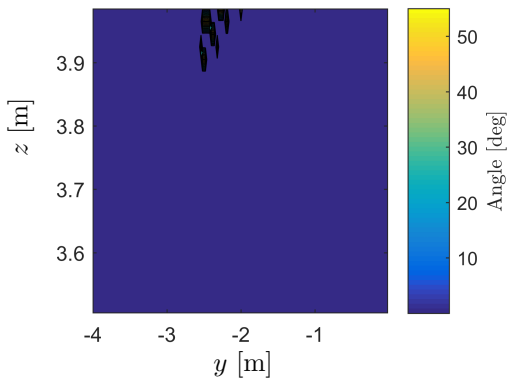
In this subsection, the simulation was performed with a diagonal motion within the feasible area of the robot workspace, as defined in the previous subsection. Three different platform speeds, of 1, 2 and 3 m/s were simulated with a platform trajectory starting from $\boldsymbol{x}_a = [0, -4, 0.465]^{\mathrm{T}}\mathrm{m}$ and proceeding to $\boldsymbol{x}_b = [0, 4, 3.5]^{\mathrm{T}}\mathrm{m}$. The velocity effect and the beam diameter effect have been taken into account in the mathematical model with the purpose of imitating a real system.

There are two different simulation results that will be shown in this section:

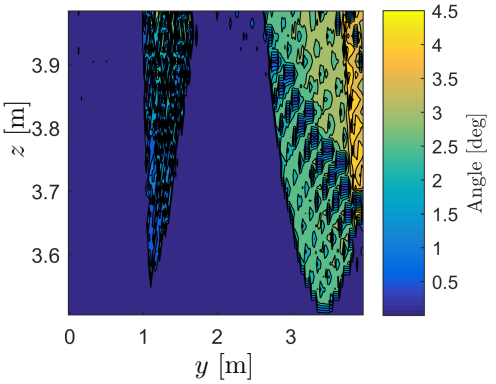
[1]Excluding the pitch angle, because the pitch angle is obtained from the IMU sensor.



a. Workspace $0.5\,\mathrm{m} \leq z \leq 3.5\,\mathrm{m}$



b. Workspace $3.5\,\mathrm{m} \leq z \leq 4\,\mathrm{m}$
$-4\,\mathrm{m} \leq y \leq 0\,\mathrm{m}$



c. Workspace $3.5\,\mathrm{m} < z \leq 4\,\mathrm{m}$
$0\,\mathrm{m} \leq y \leq 4\,\mathrm{m}$

Figure 5.10: Yaw angle.

1. simulation result without velocity compensation, and
2. simulation result with velocity compensation.

The simulation result without velocity compensation is depicted in Fig. 5.11. The absolute error of the x-component for the simulation with platform speeds of 1 and 2 $^{\mathrm{m}}/_{\mathrm{s}}$ are less than 2 cm and relatively similar. If the platform speed is increased to 3 $^{\mathrm{m}}/_{\mathrm{s}}$, the absolute error becomes quite higher, $z > 2.5\,\mathrm{m}$. This is reasonable because only a few points intersect with the two upper reflectors at the right side (denoted by reflector numbers 1 and 2 in Fig. 5.12) at that platform position. Moreover, the direction of the rotation of the scan also plays an important role in the result, as shown in Fig. 5.4. It can be seen in the same figure that the measured lengths of reflectors numbers 1 & 2 during the platform motion are almost similar with that under the stationary condition. However, the measured lengths of reflectors 3 & 4 during the motion are less than when stationary.

It still remains to discuss the simulation of a platform motion with speed 3 $^{\mathrm{m}}/_{\mathrm{s}}$. When $y \leq 0\,\mathrm{m}$ and $z > 2\,\mathrm{m}$, the proposed algorithm to estimate the x-component chooses the left corner as the reference for calculation since the horizontal distance from the platform to the left side reflector is closer than to the right side reflector. This has a disadvantage due to the deviation of the measured length of reflectors 3 & 4, as stated in the paragraph above. In this case, selecting the right corner as the reference also will not give a better result due to the deviation of the measured length of the reflector number 1 as stated in the paragraph above. In summary, the proposed algorithm to estimate the x-component during platform motion without velocity compensation works better for the platform speeds 1 & 2 m.

The platform speed also influences the absolute error of the y- and z-components, as depicted in Fig. 5.11.d. The maximum absolute errors of the y- and z-components are circa 2 cm and 1 cm, respectively, which occurs at a platform speed of 3 $^{\mathrm{m}}/_{\mathrm{s}}$. In summary, Fig. 5.11.d shows that the faster the platform speed, the higher the absolute error.

The determined platform orientations are displayed in Fig. 5.11.b and Fig. 5.11.c. The deviation of both angles is close to 0, which is satisfactory for this application.

Up to now, the simulation results for the platform motion without velocity compensation have been discussed. In the following paragraph, the similar simulation with velocity compensation will be discussed.

The results of the platform motion with velocity compensation is depicted in Fig. 5.13. The absolute error of the x-component is relatively similar for any platform speed when $0.465\,\mathrm{m} \leq z \leq 2.74\,\mathrm{m}$. Then, the error increases moderately for $2.75\,\mathrm{m} < z < 3.5\,\mathrm{m}$. Within this area, the velocity compensation algorithm increases the error instead of reducing it. In comparison with the previous result, there is little difference for $0.465\,\mathrm{m} \leq z \leq 2.74\,\mathrm{m}$. To summarize, the velocity

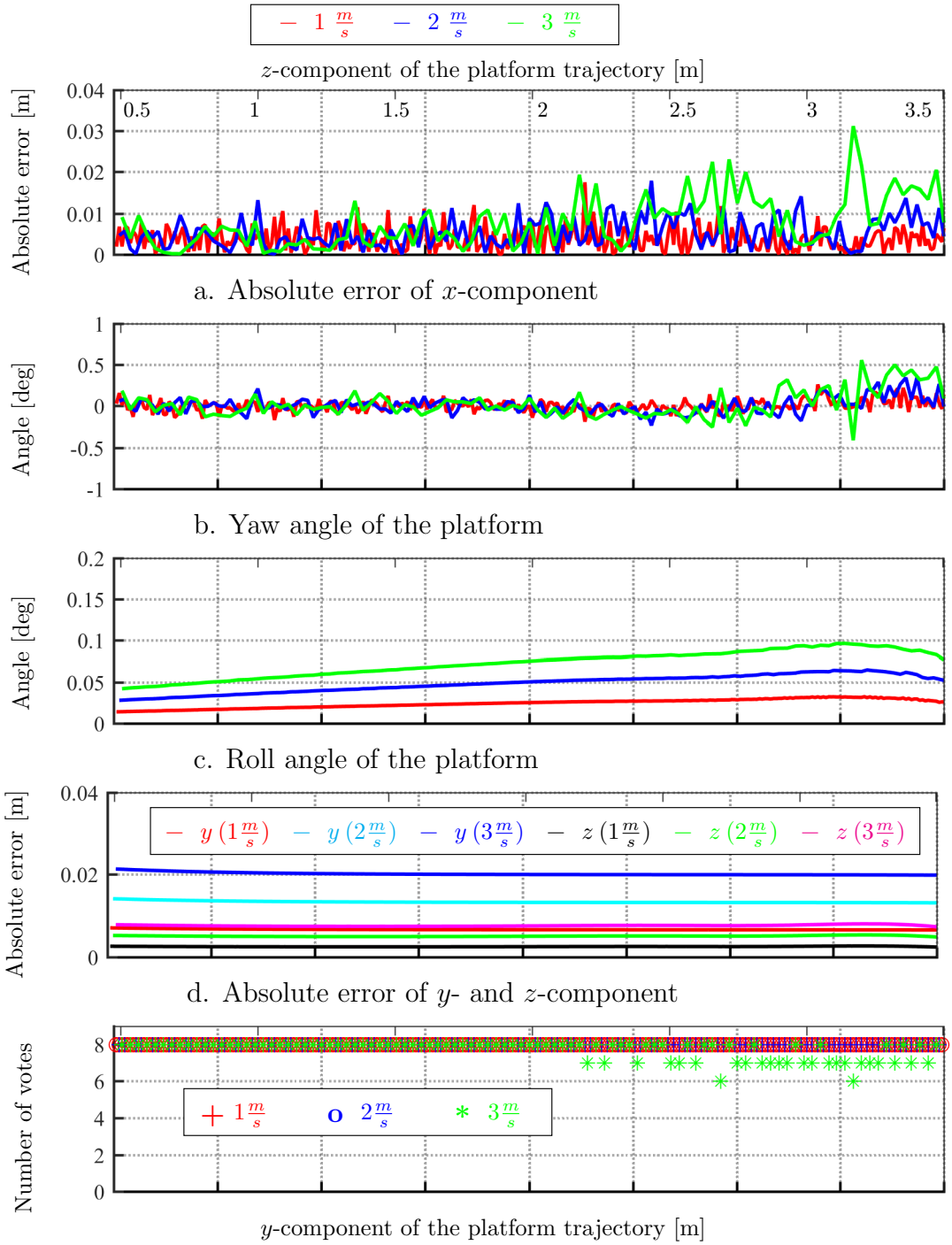


Figure 5.11: Simulation result during the platform motion without velocity compensation.

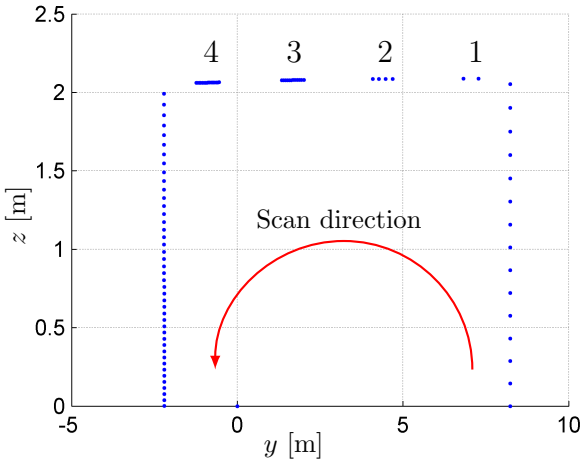


Figure 5.12: Scan results during platform motion with speed of 3 $^{\mathrm{m}}/_{\mathrm{s}}$.

compensation works better in the determination of the x-component for the region $z \leq 2.74\,\mathrm{m}$.

The yaw angle is relatively similar to the previous result between $\pm 1\,^{\circ}$. The velocity compensation algorithm works perfectly in canceling the velocity effect for the roll angle, y-, and z-component. The results are similar to the desired value.

Finally, some important points can be concluded as a recommendation for the experiment:

1. regardless of the velocity compensation algorithm, the absolute error of the x-component is relatively similar for any platform speed when $0.465\,\mathrm{m} \leq z \leq 2.74\,\mathrm{m}$. It is recommended not to compensate the velocity effect for the rest of the region in order to achieve a better estimated x-component.
2. There is a small difference in the estimated yaw angle of the proposed algorithm with and without velocity compensation.
3. The velocity compensation works perfectly to eliminate the velocity effect on the estimated roll angle, the y-, and the z-component.

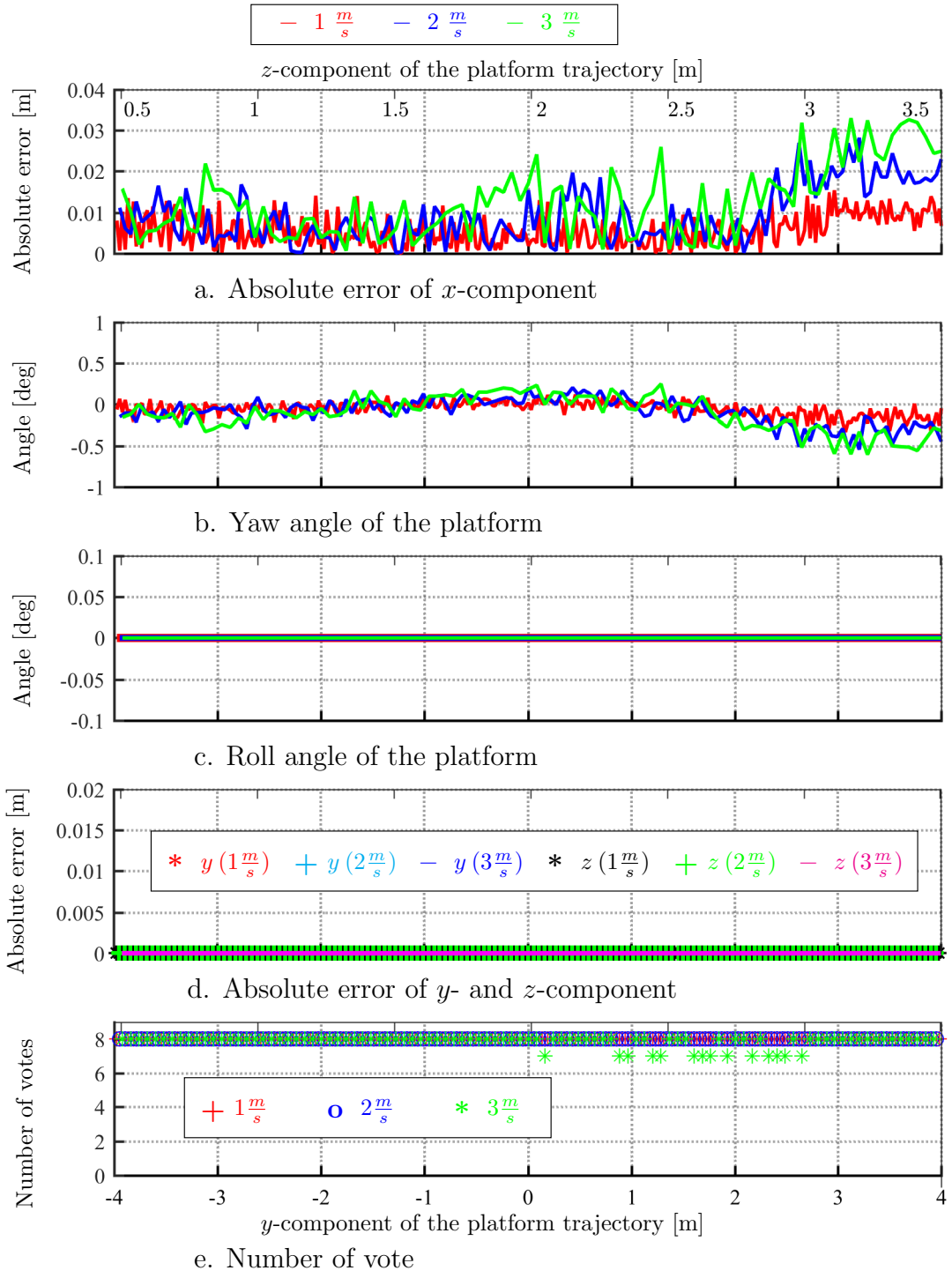


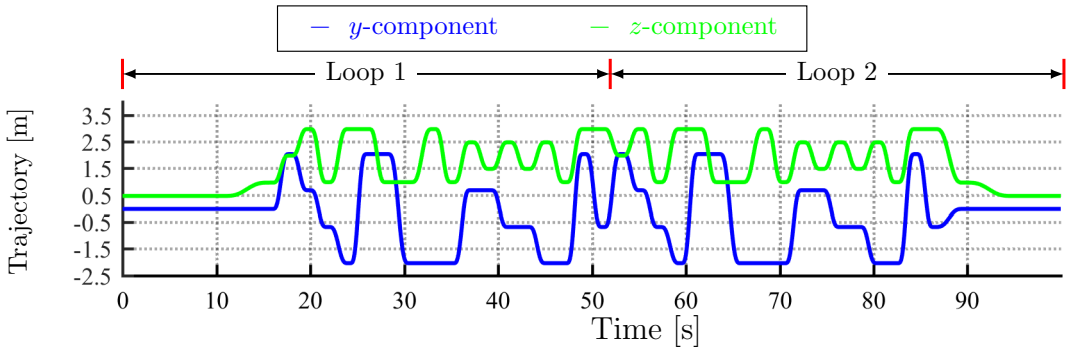
Figure 5.13: Simulation results during the platform motion with velocity compensation.

5.2 Experimental results

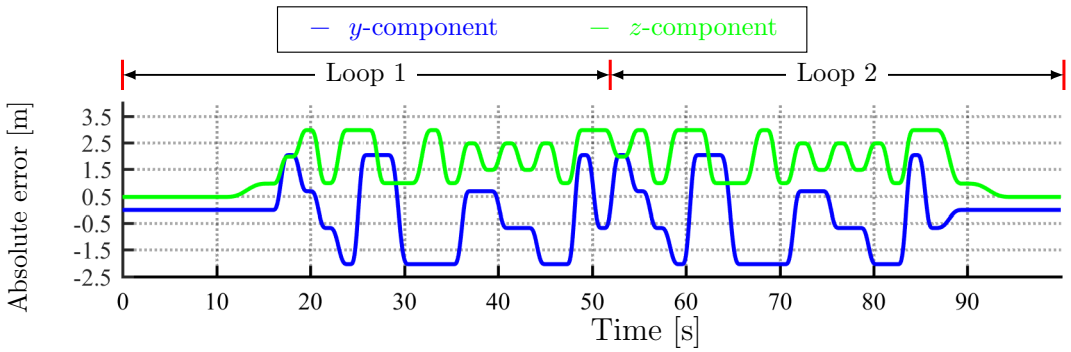
The recommendation in the previous section was taken into account for the experiment. The first important step in the experiment was the selection of a reliable interface for the data transfer by comparing the measurement results acquired from both interfaces. For purposes of simplification, a loop of motion is defined for a set of point to point motions which generates a smooth platform trajectory.

5.2.1 Comparison of serial and ethernet communication ports

Theoretically, the serial port based system architecture in Fig. 2.11 should be more reliable than the Ethernet based system in Fig. 2.14, because all the hardware and software of the serial port based system runs under a real time system with a constant frequency. But the PC104 has a resource limitation for the application in this work. The main problem is CPU overload, even after some optimization in the programming of the proposed method. The CPU is only able to compute the method proposed in Section 4.4 until step 4, with the result as depicted in Fig. 5.14. Without completing all the steps in the proposed method, the result in



a. Platform trajectory [m]

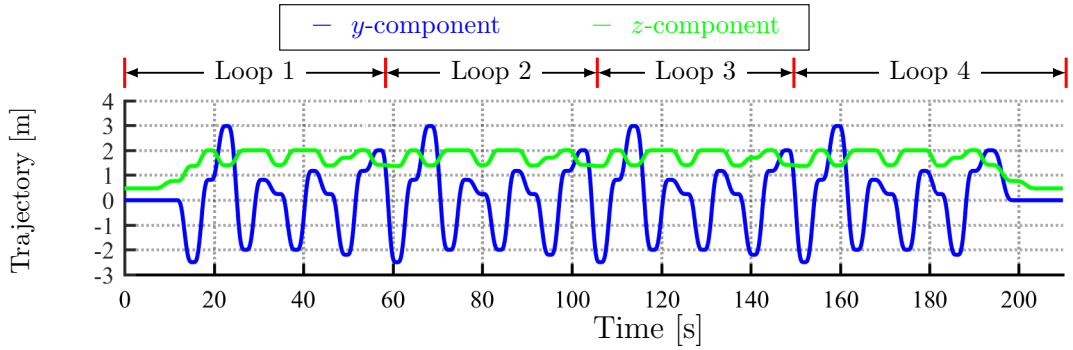


b. Absolute error of the platform trajectory [m]

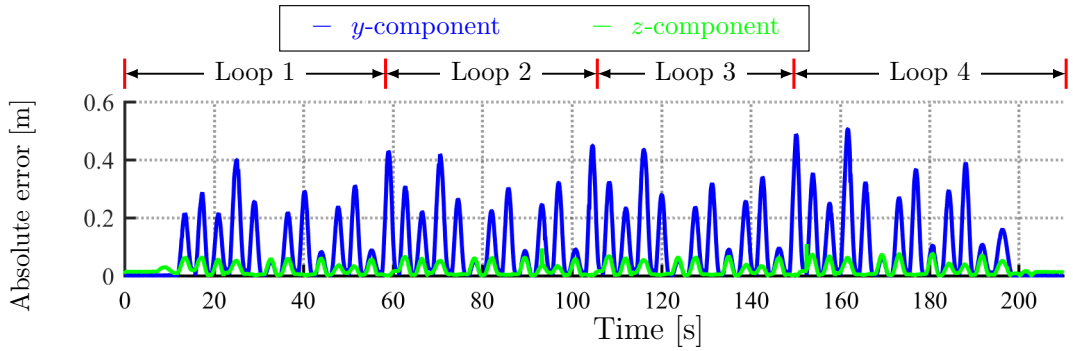
Figure 5.14: Measurement result using serial port.

Fig. 5.14.b is very noisy due to the drawback of the Hough Transformation. The drawback of the Hough Transformation has been mentioned in Section 4.1.

Meanwhile, the measurement result from the Ethernet based system is shown in Fig. 5.15. The absolute error fluctuations of each loop are relatively similar. The



a. Platform trajectory [m]



b. Absolute error

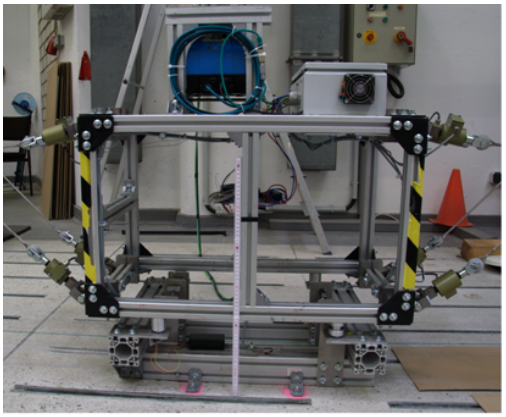
Figure 5.15: Measurement result using Ethernet port.

graph looks smoother than in Fig. 5.14.

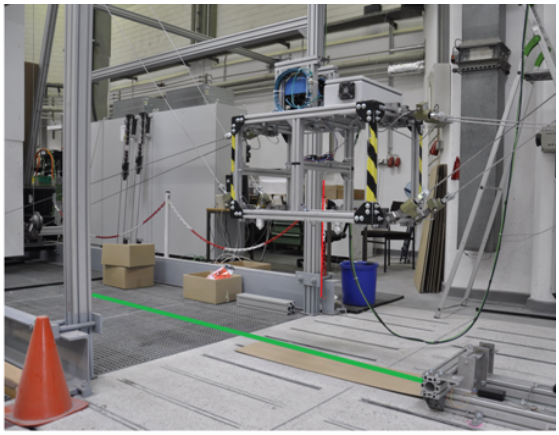
In general, the Ethernet based system outputs better results. Due to the peripherals limitation in serial port based system architecture, the Ethernet based system architecture is chosen to transmit the measurement data to the robot controller system.

5.2.2 Validation of the proposed method with direct measurements

The proposed method must be validated with the other measurement method for justification. The measurement setup to measure the platform position is depicted in Fig. 5.16, and described in the following:

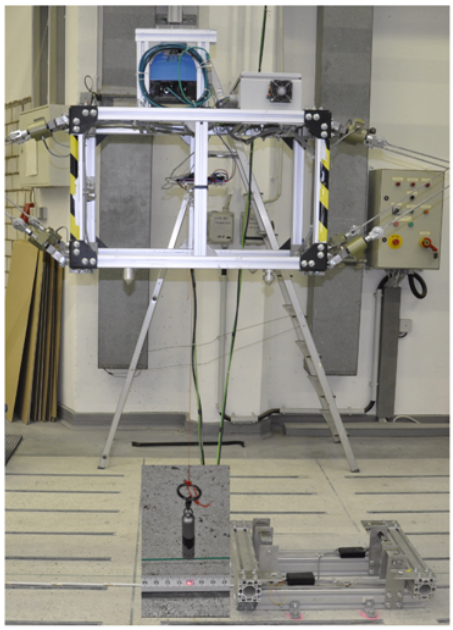


a. z-component measurement



b. Green rope along y-axis

c. Laser pointing on the scale



d. y-component measurement

Figure 5.16: Measurement setup.

- x-component measurement. A rope is stretched from the left to right frame along the y-axis of the robot's inertial system at $x = 0\,\text{m}$ and $z = 1\,\text{cm}$, as depicted in Fig. 5.16.b. A ruler is placed perpendicularly to the green rope under the laser pointer. The distance from the rope to the laser pointer is the x-component of the platform's position vector.
- y-component measurement. The ruler is placed parallel to the green rope. The distance from the robot's inertial system to the laser pointer is the y-component of the platform's position vector.
- z-component measurement. The distance from the floor ($z = 0\,\text{m}$) to the platform's coordinate system is measured by a ruler, as depicted in Fig. 5.16.a.

The data were collected while the platform was in the static (stationary) condition. Two types of experiments were conducted, as follows:

1. The y-component constant at 0.8 m while the z-component varied from 0.8 m to 1.9 m with a discretization of 0.1 m. The x- and z-components were measured and the results are shown in Fig. 5.17.a.
2. The z-component constant at 1.5 m while the y-component was varied from 0 m to 1.8 m with a discretization of 0.1 m. The x- and y-components were measured and the results are shown in Fig. 5.17.b.

The results are presented in terms of the absolute error. The legend "determined" and "measured" mean the result from the proposed method and the measured value using the ruler, respectively.

It can be seen in Fig. 5.17 that the absolute error of the y- and z-components from both measurement methods almost coincide. Meanwhile, the difference of the absolute error of x-component from both measurement methods is less than 1 cm, which is still within the algorithm error depicted in Fig. 5.8. According to the experimental results, it can be concluded that the proposed method is valid.

5.2.3 Platform position deviation

After the validation, the proposed method was ready to be applied to measure the platform pose. The platform was placed within the robot workspace at several poses, as illustrated in Fig. 5.18. The exact platform position as illustrated in Fig. 5.18 is given in Table 5.1, which is referred to as the desired pose. In the same table, the measurement results, presented as the absolute error of the platform pose, are also shown. The maximum value of each pose component is in boldface.

The measurement results show that the maximum absolute errors of the x- and y-components are 3.2 cm and 3 mm, respectively. However, the z-component attracts more attention than the other component because the platform deviation increases significantly with increasing z-component. One of the possible reasons for this

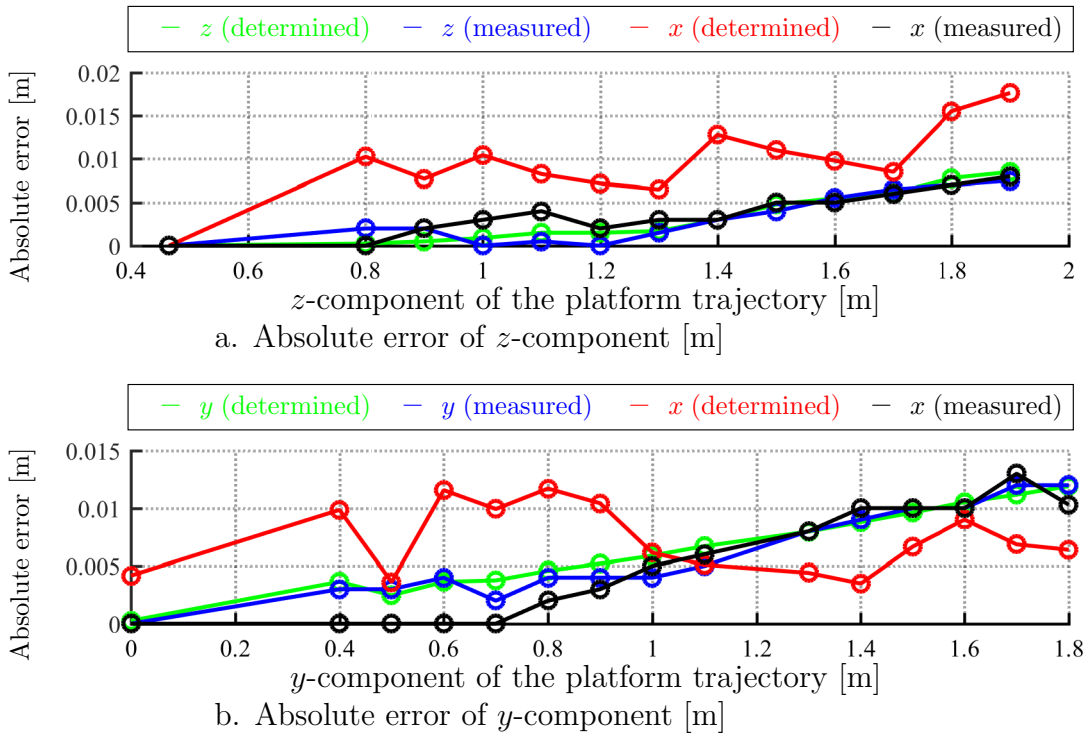


Figure 5.17: Comparison of the proposed method and direct measurement.

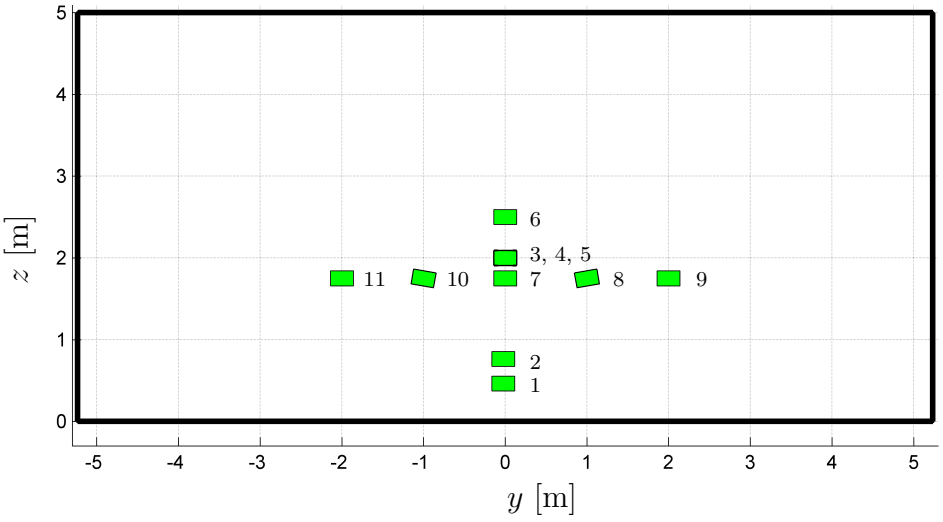


Figure 5.18: Platform pose measurement within the workspace.

Table 5.1: Comparison between desired and measured platform poses before parameter change.

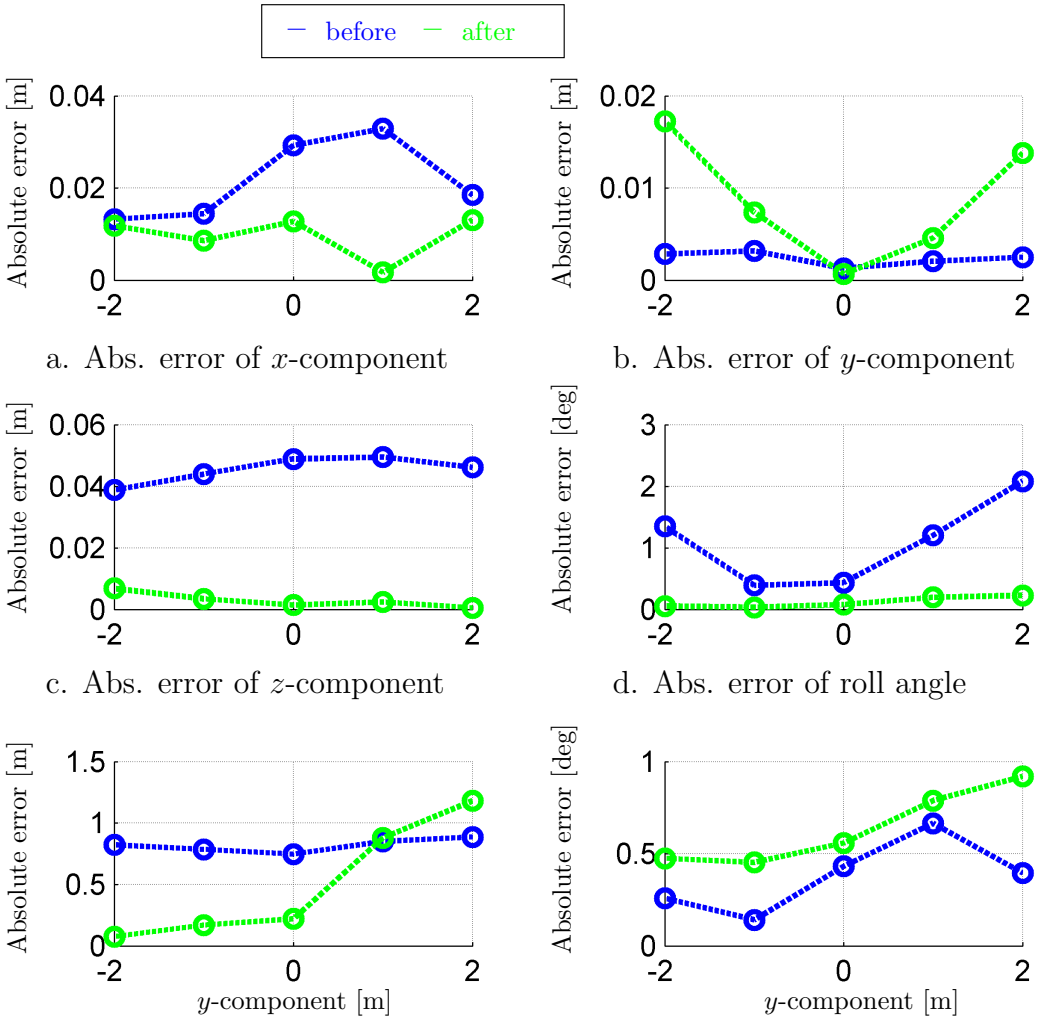
No.	**Desired pose ([m] and [deg])**						**Absolute error ([m] and [deg])**					
	x	y	z	φ	ϑ	ψ	x	y	z	φ	ϑ	ψ
1	-0.027	-0.022	0.465	0	0	0	0.012	0.001	0.004	0.37	0.34	0.05
2	-0.027	-0.022	0.765	0	0	0	0.023	0.001	0.008	0.38	0.57	0.30
3	0	0	2	0	0	0	0.03	0.001	0.062	0.42	0.97	0.65
4	0	0	2	2	0	0	0.026	0	0.061	0.30	**1.03**	0.55
5	0	0	2	-2	0	0	0.018	0.002	0.62	0.51	0.93	0.55
6	0	0	2.5	0	0	0	0.013	0	**0.085**	0.33	0.86	0.28
7	0	0	1.75	0	0	0	0.017	0.002	0.05	0.44	0.86	0.50
8	0	1	1.75	2	0	0	0.01	0.002	0.05	1.21	0.97	**0.65**
9	0	2	1.75	0	0	0	**0.032**	0.002	0.046	**2.1**	0.98	0.47
10	0	-1	1.75	-2	0	0	0	0.003	0.045	0.42	0.86	0.21
11	0	-2	1.75	0	0	0	0.011	0.003	0.039	1.38	0.92	0.26

deviation is the defined kinematic parameters in the robot controller not being similar to that of the real system. In more detail, the height of the upper pulley in the controller could be different from the real system. This is reasonable because the measured z-component of the platform is less than the desired one. To expose the influence of the upper pulley height in the kinematic parameters of the robot controller on the platform positioning, the heights of the upper pulleys in the kinematic parameters were increased a few centimeters. Then, the data from the similar platform pose as Table 5.1 were collected an d are presented in Table 5.2.

For a better view of before and after the parameters change, Table 5.1 and Table 5.2 are shown in two graphs: horizontal and vertical graphs. The horizontal graph consists of the platform pose numbers 11, 10, 7, 8 and 9 while the vertical graph consists of platform poses 1, 2, 7, 3 and 6.

The comparison of the absolute error before and after changing the kinematic parameters for the horizontal and the vertical measurements is depicted in Fig. 5.19, and Fig. 5.20 respectively.

Updating the kinematic parameters influence the platform positioning significantly, as depicted in Fig. 5.19 and 5.20. Some absolute errors decrease and some others slightly increase. Overall, the platform positioning after the parameters update is better than before. The better result after updating the kinematic parameters confirms the statement in Sec. 1.2, where the defined kinematic parameters in the controller must be very close if not match with the actual on the robot. The kinematic parameters of the robot are required in the robot controller as depicted



a. Abs. error of x-component

b. Abs. error of y-component

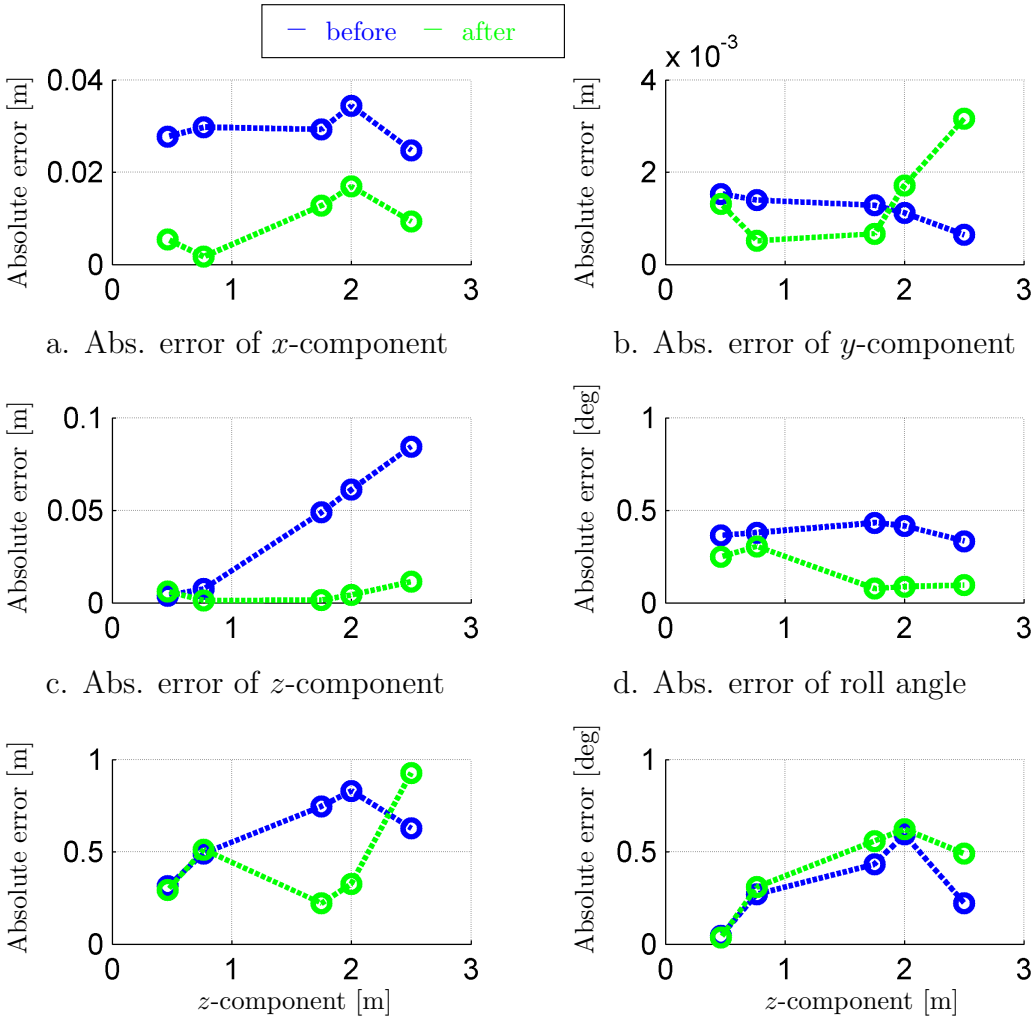
c. Abs. error of z-component

d. Abs. error of roll angle

e. Abs. error pitch angle

f. Abs. error of yaw angle

Figure 5.19: Comparison of before and after calibration, horizontal.



a. Abs. error of x-component

b. Abs. error of y-component

c. Abs. error of z-component

d. Abs. error of roll angle

e. Abs. error pitch angle

f. Abs. error of yaw angle

Figure 5.20: Comparison of before and after calibration, vertical.

Table 5.2: Comparison between desired and measured platform pose after parameter change.

No.	**Desired pose ([m] and [deg])**						**Absolute error ([m] and [deg])**					
	x	y	z	φ	ϑ	ψ	x	y	z	φ	ϑ	ψ
1	-0.027	-0.022	0.465	0	0	0	0.005	0.001	0.006	0.25	0.29	0.04
2	-0.027	-0.022	0.765	0	0	0	0.0017	0.00	0.0013	0.30	0.51	0.30
3	0	0	2	0	0	0	**0.017**	0.001	0.0043	0.08	0.32	0.62
4	0	0	2	2	0	0	0.0016	0.0024	0.0024	0.25	0.49	0.54
5	0	0	2	-2	0	0	0.0145	0.0010	0.0026	0.08	0.24	0.74
6	0	0	2.5	0	0	0	0.0093	0.0032	**0.011**	0.09	0.92	0.49
7	0	0	1.75	0	0	0	0.0128	0	0.0015	0.07	0.22	0.55
8	0	1	1.75	2	0	0	0.0017	0.0046	0.0024	0.19	0.88	0.78
9	0	2	1.75	0	0	0	0.0131	0.0138	0.00	0.22	1.18	0.91
10	0	-1	1.75	-2	0	0	0.0086	0.0074	0.0034	0.03	0.17	0.45
11	0	-2	1.75	0	0	0	0.011	**0.017**	0.006	0.05	0.07	0.47

in Fig. 1.15. The updated kinematic parameters were used for the experiment whose results are presented the following subsections.

5.2.4 Measurement delay and its compensation

Up to the present time, the experiment was conducted with the platform stationary. Before a dynamic measurement can be performed, the measurement delay must be compensated. The measurement delay has been mentioned in Sect. 2.4.

The measurement delay is determined from the time shift between the desired and measured translational component of the platform trajectory. In this study, the measurement delay is determined from the y-component because the platform has a longer trajectory in this direction than in the others. An example of the y-component of the platform trajectory during the motion and the total time delay is displayed in Figs 5.21.a. 5.21.d., respectively.

The area inside the red circle is enlarged to show the time shift of the desired and measured trajectory clearly. Without time compensation, the absolute error of the trajectory is depicted in Fig. 5.21.b., where there are spikes during the platform motion. The spikes increase if the measured trajectory is compensated with the total time delay. Meanwhile, the compensation of the measurement result with the estimated delay from cross correlation as explained in Section 2.4 decreases the absolute error as depicted in Fig. 5.21.c., but the spike still exists. To cancel the spike completely, the estimated delay must be adjusted manually by increasing or decreasing the time estimated delay. If the spike is eliminated, as shown in Fig. 5.21.c., the adjusted time delay is considered as the real measurement delay.

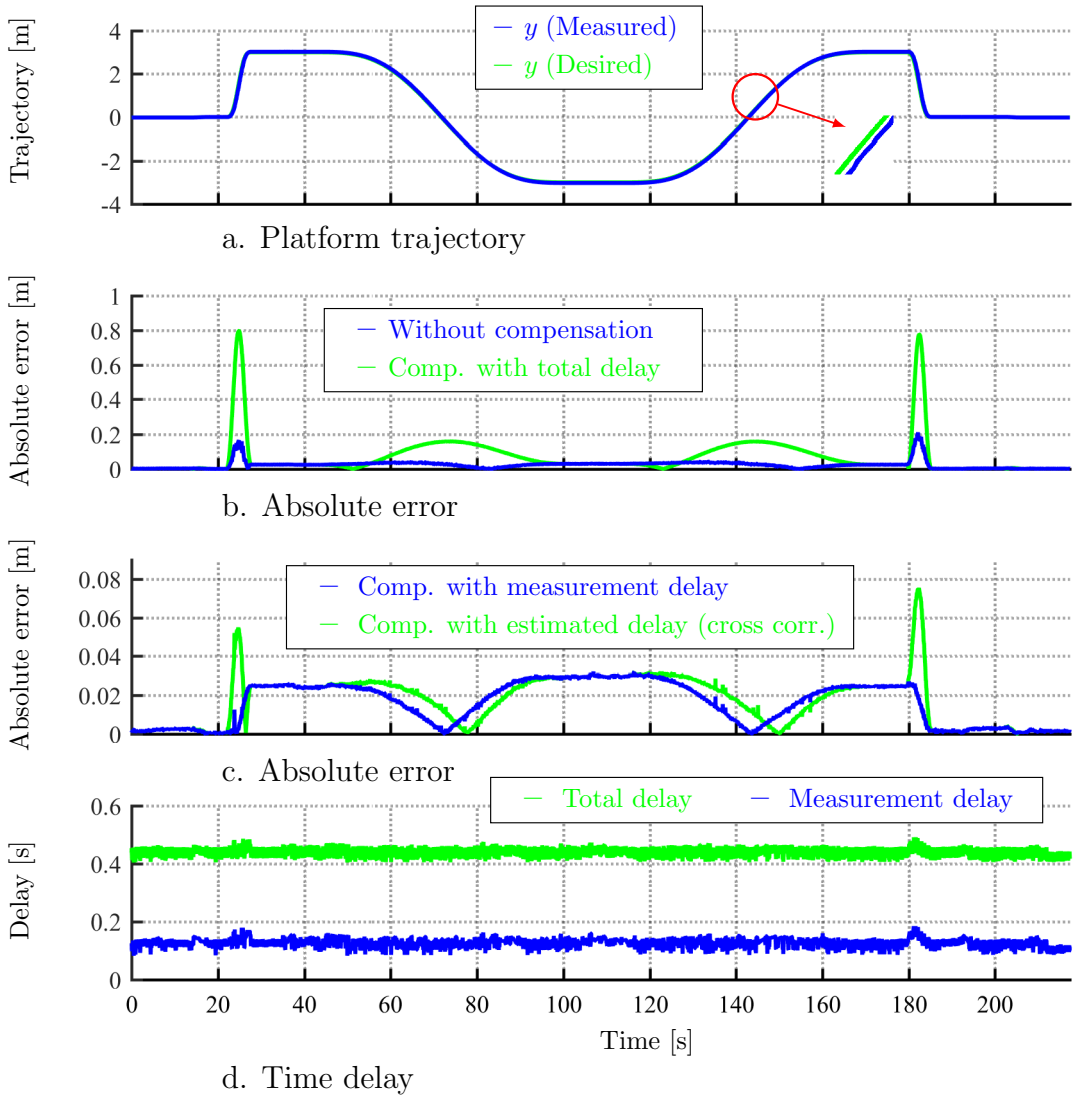


Figure 5.21: The absolute error of the platform trajectory and time delay.

The comparison of the total time delay (from the laser scanner) with the real measurement delay is depicted in Fig. 5.21.d.

5.2.5 Measurement results for low and high speed

The delay compensation as mentioned in the previous subsection allows an experiment under dynamic conditions. For application to a real system, the proposed method must be able to provide the measurement result (determined platform pose) for both low and high speeds of the platform motion. Low and high speeds are defined for platform motion as a maximum speed of circa 0.2 m/s and 3.13 m/s, respectively.

Theoretically, the robot is designed with a maximum platform speed of 5 m/s. However the maximum platform speed is limited to circa 3 m/s for safety reasons. To prove the reliability of the proposed method, the experimental results with low and high platform speeds will be shown in the following paragraphs.

The experimental results for a platform trajectory similar to that in Fig. 5.21 and a platform speed of circa 0.2 m/s are shown in Fig. 5.22 and Fig. 5.23. The absolute

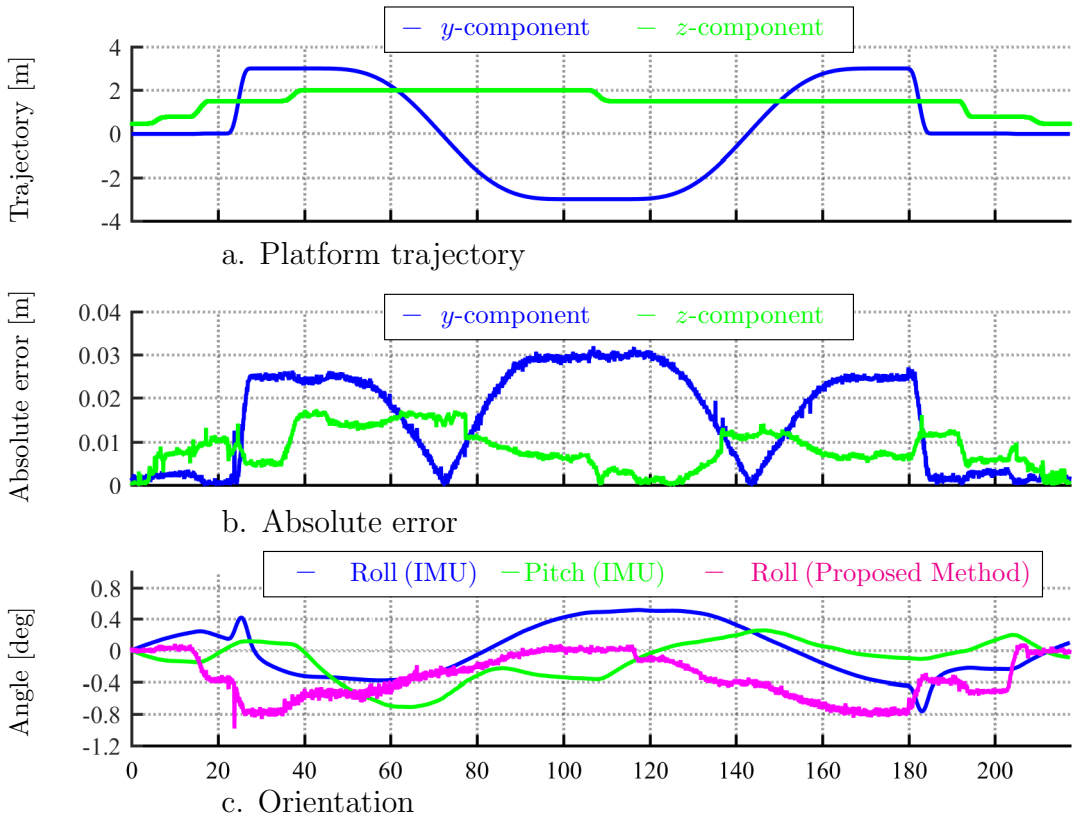


Figure 5.22: Absolute error of the platform pose and the orientation.

error of the y-component depends on the y-component of the platform position as

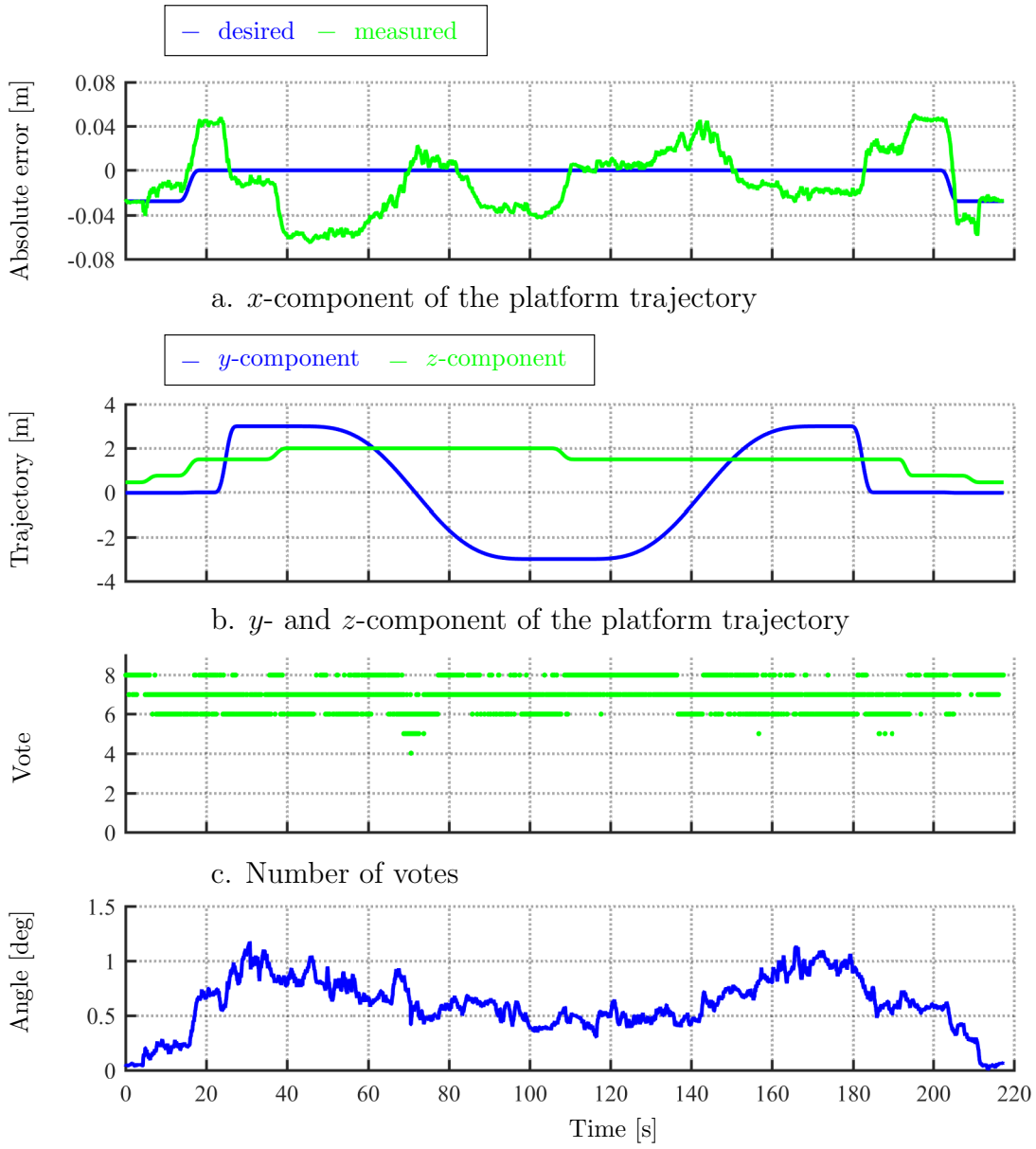


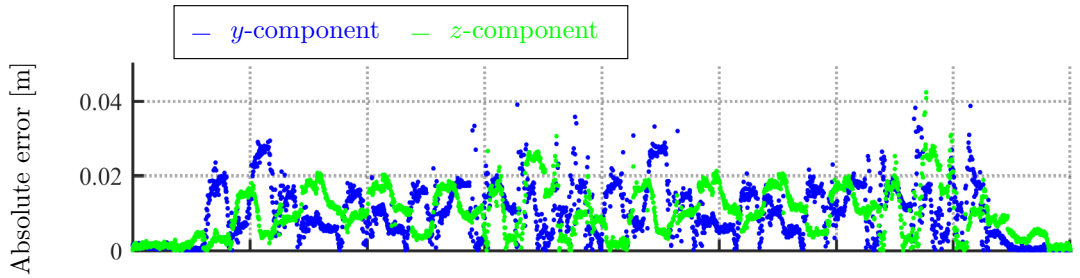
Figure 5.23: x-component of the platform position and yaw angle.

depicted in Fig. 5.22.b. The absolute error increases linearly with the y-component of the platform position. Meanwhile, the absolute error of the z-component behaves like the y-component but with an irregular deviation at certain poses. One of the problems during the experiment with a low platform speed was the friction in the rotating parts, especially in the motor gearbox on the platform. Due to Coulombs friction, the platform motion is not smooth at low speeds. The platform vibrates with a low frequency in the z-direction because the controller compensates the high friction all the time during the platform motion. Meanwhile, the absolute error of the x-component is higher than the other translational components of the platform pose. This is reasonable because the algorithm itself has an internal error, as mentioned in Section 5.1. In addition, the pulley position does not allow the controller to generate a high force in the x-direction to increase the platform's stiffness. The low force in the x-direction allows the platform to vibrate slowly in the x-direction. Increasing the wire force could be a solution but the controller has a so called emergency stop, which is active for instance if the wire fore exceeds the defined limit.

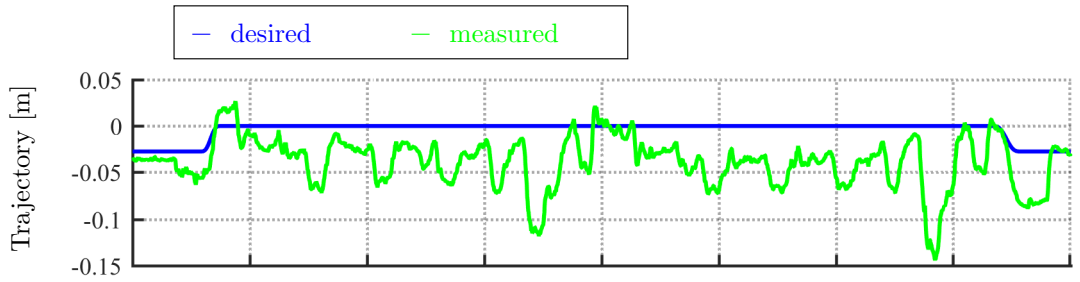
The orientation of the platform during the motion is less than $\pm 1°$. The measured roll angle from the IMU and the proposed method have the same pattern (Fig. 5.22.c.). In this study, the yaw angle is not compared for the validation. But the estimated yaw angle is reasonable because the laser scanner always intersects with the reflector, which has been modeled in Section 3.6. Furthermore, Fig. 5.23.c shows that the number of votes is greater than or equal to 5 (only one point has less than 5).

By now, the low speed experiment has been presented and discussed. The proposed method was then ready to be applied to the high speed experiment. A point to point trajectory with two loops is chosen. The second loop was started at time 80 s. The platform had a maximum speed of $3.13\,^{\mathrm{m}}/_{\mathrm{s}}$. The experimental results are shown in Fig. 5.24. The absolute error of the y- and z-components have a maximum value of circa 4 cm. There is no spike which is typical of a measurement delay as in Fig. 5.21. The absolute error fluctuates at several platform positions due to platform vibration or increasing/decreasing velocity[2]. Meanwhile, the x-component fluctuates moderately. At a specific platform pose, the deviation of the x-component from the desired and the pitch angle increase a lot in comparison with other poses. In terms of the number of votes, the measured x-component is trusted because the number of votes is mainly larger than 5.

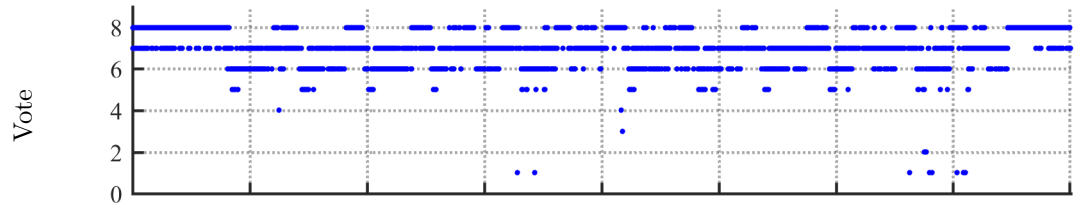
[2]The compensation algorithm works perfectly for a constant velocity.



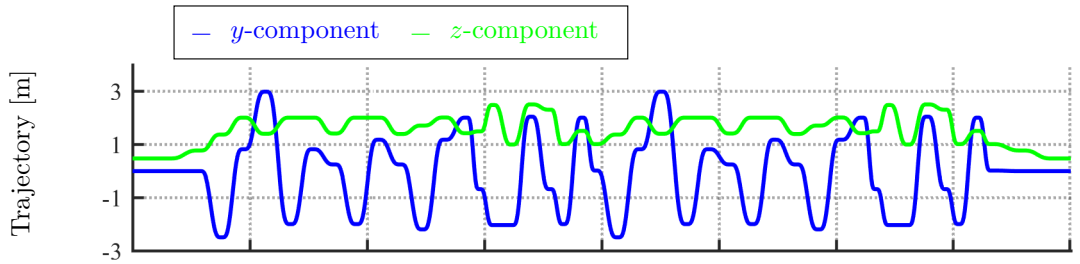
a. Absolute error of the platform trajectory



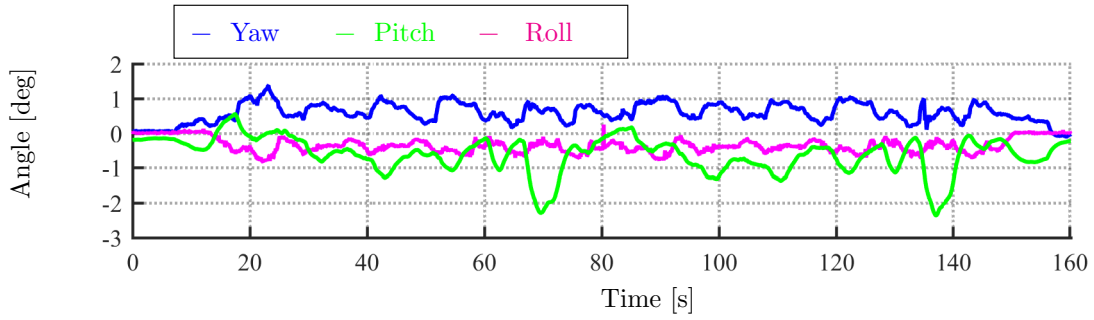
b. x-component of the platform trajectory



c. Number of votes



d. Platform trajectory



e. Platform orientation

Figure 5.24: High speed experimental result.

5.3 Concluding remarks

The Ethernet based system architecture was chosen for communication and data transmission between the computer and the laser scanner through the EtherCAT system. In comparison with the serial port based system architecture, the Ethernet based system architecture is able to transmit all measurement data from the laser scanner to the module in the robot controller. The determined platform pose has been validated with the direct measurement method. Furthermore, the proposed method is proven to determine the platform pose within the specific workspace mentioned above. The velocity compensation algorithm works sufficiently well to reduce the velocity effect on the measurement data from the laser scanner.

CHAPTER 6

Conclusions, Contributions and Future Work

6.1 Conclusions

The main purpose of this work has been to study the implementation of a laser scanner as a measurement tool and the development of an algorithm to determine the platform pose of a particular prototype of a cable robot, CABLAR. The work consists of three main stages: modeling, simulation, and testing of the prototype.

The laser scanner Sick LMS500 and Inertial Measurement Unit Microstrain 3DM-GX1 were chosen as the measurement tools. An Ethernet interface was selected instead of a serial port for communication and data transmission between the laser scanner and the host computer. A software application to drive the laser as well as to synchronize the internal time of the laser scanner with that of the host computer under the C++ environment was developed. The measurement data was processed by the developed algorithm, which was also programmed under the C++ environment.

Reflectors with a special pattern were designed in order to generate unique measurement data. A mathematical model has been developed to determine three translational and two rotational components of the platform pose from 2D measurement data. A mathematical model to extract the x-component of the platform pose has been established.

The kinematic model of the CABLAR as a basis for further research and a mathematical model based on a part of the Gradient Projection Method and the properties of the laser scanner have been developed to simulate the intersection of the laser beam with the reflector within the robot's workspace. The platform velocity is also taken into account in order to imitate a real system. The simulation yielded a set of data imitating measurement data from a real system. A mathematical model to compensate the velocity effect in the measurement data from the laser scanner has been developed.

A fast Hough Transform based on a hierarchical approach has been modified in order to decrease the use of computer memory. Together with the Random Sample Consensus algorithm and Linear Least Squares, the modified Hough Transform has been combined to determine the normal parametrization of the reflectors. Furthermore, a distance-based threshold split and merger algorithm segmented the measurement data corresponding with a specific reflector. Several schemes for the conditions in a real system have been simulated in order to study the characteristics of the algorithm. There resulted several points for recommendations to be considered in experimental work.

The platform pose determined by the proposed method was validated by direct measurement. Then, the platform was placed in certain poses within the robot's workspace to collect the actual platform poses. The desired and determined poses were compared in order to benchmark the controller. The results show that the platform has a deviation of a few centimetres, and is proportional to the platform pose. The proposed algorithm was also tested to determine the pose during the platform motion. The results show that the proposed algorithm is real-time capable and able to measure the actual platform pose.

6.2 Scientific and Technical Contributions

The main contribution of this thesis is the use of a two-dimensional laser scanner in combination with a reflector with a special pattern design to determine 5 among the 6 components of the platform position. The use of a laser scanner allows the direct measurement of the platform position. In more detail, the scientific and technical contributions are elaborated as follows.

The scientific contributions of this thesis are:

1. A threshold based on the distance and the angular resolution for object segmentation from the measurement data of a laser scanner has been proposed. The proposed threshold allows the segmentation algorithm to be more robust in application.

2. A fast Hough Transform in Li et al. [1986] has been improved for the application to CABLAR. The modified Hough Transform is faster than that in Li et al. [1986].
3. A strategy based on a three straight line extraction algorithm to separate the line corresponding to the data from a set of measurement data from the laser scanner has been proposed. This strategy is effective at removing outliers and obtaining the ideal model line of a data set and in the end to obtain the normal parametrization of a straight line.
4. A reflector with a special pattern has been designed. A mathematical model based on the reflector design has been developed. The reflector design and its mathematical model have been tested on the prototype.

Meanwhile, the technical contribution of this thesis is the software to drive the Sick 2D LMS500 laser scanner. The software is written in the C++ environment based on Windows Socket Programming.

6.3 Future work

For future work, the platform pose determined by the proposed algorithm should be validated with direct measurements using a high precision measurement tool such as laser tracker. Equally important is the calibration of the kinematic parameters in the robot controller, either by a laser tracker or by self calibration.

It was mentioned in the Sec. 1.2 that one of the objectives of this study is to provide the actual platform pose as reference value. If the actual pose error exceeds the tolerance then the platform must be return to the home position for referencing the cable length. Since the robot is not calibrated correctly, the accuracy of platform positioning by the controller is still low. Due to that reason, the error elimination by can not be realized in this work since the pose error is influenced by the error from not calibrated robot. Therefore, this step should be realized in future work after the robot is calibrated.

In this work, the proposed algorithm and the robot controller ran on a PC with a two-core processor. To reduce the load on the PC, it is recommended to prepare a higher performance PC or a dedicated PC to drive the laser scanner and to process the measurement data in order to determine the platform pose.

APPENDIX A

Intersection of two vectors

Consider two direction vectors $\boldsymbol{X}_1$ and $\boldsymbol{X}_2$ located at $\boldsymbol{X}_{01}$ and $\boldsymbol{X}_{02}$ as depicted in Fig. A.1 respectively. All vectors are $\in \mathbb{R}^3$. If the direction vectors are not parallel, the intersection point $\boldsymbol{X}_{03}$ is computed by

$$\boldsymbol{X}_{01} + \lambda_1 \boldsymbol{X}_1 = \boldsymbol{X}_{02} + \lambda_2 + \boldsymbol{X}_{02} = \boldsymbol{X}_{03} \tag{A.1}$$

where λ_1 and λ_2 are scalars. Eq. A.1 can be rewritten as

$$\lambda_1 \boldsymbol{X}_1 = (\boldsymbol{X}_{02} - \boldsymbol{X}_{01}) + \lambda_2 \boldsymbol{X}_{02} \tag{A.2}$$

and cross multiplication with $\boldsymbol{X}_2$ applied to both sides to eliminate λ_2 yields

$$\lambda_1 (\boldsymbol{X}_1 \times \boldsymbol{X}_2) = (\boldsymbol{X}_{02} - \boldsymbol{X}_{01}) \times \boldsymbol{X}_2. \tag{A.3}$$

Now λ_1 is solved by taking the magnitude of each side and dividing. Insert λ_1 into Eq. A.1, resulting in the desired intersection point.

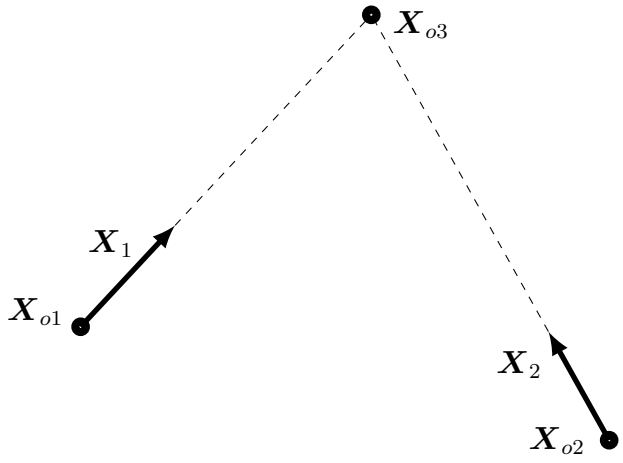


Figure A.1: Intersection point two vectors.

APPENDIX B

Linear least squares method for fitting, and the shortest distance to the origin

Consider n collinear points (x_i, y_i) where $i = 1, 2, 3, ..., n$. The equation of a straight line is

$$y = mx + c \tag{B.1}$$

where m and c are the slope and intercept, respectively, and can be obtained from dataset as follows.

$$m = \frac{\sum_{i=1}^{n} x_i y_i - \frac{\left(\sum_{i=1}^{n} x_i\right)\left(\sum_{i=1}^{n} y_i\right)}{n}}{\sum_{i=1}^{n} x_i^2 - \frac{\left(\sum_{i=1}^{n} x_i\right)^2}{n}} \tag{B.2}$$

$$c = \frac{\sum_{i=1}^{n} y_i - \left(m \sum_{i=1}^{n} x_i\right)}{n}. \tag{B.3}$$

Eq. B.1 can be rewritten as

$$0 = mx - ay + c \tag{B.4}$$

where a is the constant of component y. Then the normal vector $\boldsymbol{n}$ of the fitted line is

$$\boldsymbol{n} = [m, -a]^T \tag{B.5}$$
$$\boldsymbol{n} = [-m, a]^T . \tag{B.6}$$

Eq. B.6 is normalized to obtain the unit vector by

$$\widetilde{\boldsymbol{n}} = \frac{\boldsymbol{n}}{\|\boldsymbol{n}\|}. \tag{B.7}$$

The shortest distance from the straight line to the origin is computed by

$$d = \frac{|c|}{\|\boldsymbol{n}\|} \tag{B.8}$$

and the normal parameterization of the straight line in the Cartesian coordinate system is given by

$$\boldsymbol{X} = \widetilde{\boldsymbol{n}} d. \tag{B.9}$$

The perpendicular vector of $\boldsymbol{X}$ is

$$\boldsymbol{n}_X = [-X_y, X_x]^T \tag{B.10}$$
$$. \tag{B.11}$$

where $-X_y$ and X_x are the x- and y-components of $\boldsymbol{X}$.

APPENDIX C

Script to control Unigate CL-EtherCAT

```
1  ScriptName "Template EtherCAT 8 Byte I/O"
2  ScriptAuthor "(Rudi Kurniawan) "
3  ScriptVersion "V 1.1"
4
5  // Transparenter data exchange RS-interface <-> fieldbus
6  // System
7  SetErrorHandler         ( :AddressLabe1 ) ;
8  // see Help - Index - Inhalt - Appendix - Return Codes !
9  var bErrorHandler    : byte ;
10 SetSystemErrorHandler ( :SysAddressLabe1 ) ;
11 var bSysErrorHandler     : byte ;
12 // Bus
13 var aBusInBuf       : BUFFER [19];
14 var aBusOutBuf      : BUFFER [1024];
15 var wBusInSize      : word ;
16 var wBusOutSize     : word ;
17 var bBusInSize      : byte ;
18 var bBusOutSize     : byte ;
19 var bBusChanged     : byte ;
20 var LBusState       : Long ;
21 var bNewBusData     : byte ;
22 // General
23 var wRxLen          : word ;
24 var w16             : word ;    moveconst ( w16, 16 ) ;
```

```
25 var TimeOut        : word ;    moveconst ( TimeOut,4) ;
26 var b0             : byte ;    moveconst ( b0, 0 ) ;
27 var b1             : byte ;    moveconst ( b1, 1 ) ;
28
29 //********** init RS interface ****************************
30 Set ( RS_Type , RS422 ) ;
31 Set ( Databits , 8 ) ;
32 Set ( Stopbits , 1 ) ;
33 Set ( Baudrate , 500000 ) ;
34 //********** init Fieldbus interface **********************
35 var L_DataExch      : Long ;    moveconst ( L_DataExch, 0xE0)
      ;
36 //SetSerialInBufLen ( Buftest[0] , w510 ) ;
37 call :InitFieldbus;
38 //*********** start fieldbus ******************************
39 BusStart ;
40 :LoopBusState;
41  Get ( ReadBusState , LBusState ) ;
42  if LBusState less L_DataExch then :LoopBusState;
43
44 //call :opFBsetAfterBS;
45 // optional fieldbus settings After BusStart
46  Get ( BusInputLen16   , wBusInSize ) ;
47  Get ( BusOutputLen16 , wBusOutSize ) ;
48 //******************** main ********************************
49 :between;
50
51       Get ( RSInputCharacter16 , wRxLen ) ;
52       if wRxLen greater w16 then :next;
53 jump :between;
54 :next;
55       ReceiveSomeCharRS ( TimeOut , aBusOutBuf[0] , wRxLen)
            ;
56       WriteBus ( aBusOutBuf[0] , wBusOutSize ) ;
57 jump :between;
58 //
   ------------------------------------------------------------

59 :AddressLabe1 ;
60  Get ( ErrorCode , bErrorHandler ) ;
61  // see Help - Index - Inhalt - Appendix - Return Codes !
62 // TBD user evaluation for the value of Errorcode
63 return ;
64 :SysAddressLabe1;
```

```
65   Get ( SystemError , bSysErrorHandler ) ;
66   // see manual Errorhandling
67  return;
68  //
    ------------------------------------------------------------------------

69  :InitFieldbus;
70  // bus protocol : EtherCAT
71  // bus behaviour: event
72  // bus-data len : depending on object
73     Set ( BusInputSize , 19 ) ;
74     Set ( BusOutputSize , 60 ) ;
75  // TBD
76  return;
```

APPENDIX D

Socket Programming

```
1
2  //-------------------------------------------------------
3  //Beckhoff ADS Schnittstellen variablen
4  AdsVersion* pDLLVersion;     //ADS Versionshinweis
5  long            nErr, nPort;
6  AmsAddr         Addr;
7  PAmsAddr        pAddr = &Addr;
8  unsigned long   lHdlVar;
9  int             nIndex;
10 USHORT          nAdsState;
11 USHORT          nDeviceState;
12 char                szVar[] = { "MAIN.Sickvalue" };
13 int i;
14 char sendbuf[256];
15 char recvbuf[7000];
16 u_char flag_header_data[14] = { "LMDscandata 1" };
17 float ii=0;
18
19 int main()
20 {
21     int i; int j;
22     int rc;
23     int *rcv;
24     int size_buff = 7000;
```

```
25      float valid_result [L_VALID_RESULT][2]={};
26      WSADATA wsaData;
27      SOCKET sConnect;
28      sockaddr_in conpar;
29
30      // ws2_32.dll aktivieren
31      rc = WSAStartup(MAKEWORD(2, 0), &wsaData);
32      if (rc == 0)
33          std::cout << "WSAStartup()\t\t successful" << endl;
34      else
35          std::cout<<"error WSAStartup():"<<WSAGetLastError()
              << endl;
36
37      // socket einrichten
38      sConnect = socket(AF_INET, SOCK_STREAM, 0);
39      if (sConnect != INVALID_SOCKET)
40          std::cout << "socket()  \t\t successful" << endl;
41      else
42          std::cout << "error socket(): " << WSAGetLastError
              () << endl;
43
44      // verbindungsparameter
45      conpar.sin_addr.s_addr = inet_addr("169.254.209.81");
46      conpar.sin_family = AF_INET;
47      conpar.sin_port = htons(2111);
48      int conparlen = sizeof(conpar);
49
50      // server vor client starten, oder hier
51      // eine connect-schleife erstellen
52      rc = connect(sConnect, (struct sockaddr*)&conpar,
          conparlen);
53      if (rc != SOCKET_ERROR)
54          std::cout<<"connect() \t\t successful" << endl;
55      else
56          std::cout<<"not connected(): "<<WSAGetLastError()<<
              endl;
57
58      memset(&sendbuf, 0, sizeof(sendbuf));
59      strcpy_s(sendbuf, "\x02sEN LMDscandata 1\x03");
60
61      rc = send(sConnect, sendbuf, strlen(sendbuf), 0);
62      if (rc == SOCKET_ERROR)
63          std::cout << "error send(): " << WSAGetLastError()
              << endl;
```

```
64      std::cout << "Sick Scanner gestartet...\n";
65      memset(&sendbuf, 0, sizeof(sendbuf));
66      memset(&recvbuf, 0, sizeof(recvbuf));
67
68      //Beckhoff ADS Schnittstelle erzeugen
69      // Open communication port on the ADS router
70      nPort = AdsPortOpen();
71      nErr = AdsGetLocalAddress(pAddr);
72      if (nErr) cerr << "Error: AdsGetLocalAddress: " << nErr
            << '\n';
73      // Select Port: TwinCAT 3 PLC1 = 851
74      pAddr->port = 852;
75  //  pAddr->netId = { 134, 91, 120, 189, 1, 1 }; //AMS Net
        ID vom Cablar
76      Addr.netId.b[0]=134;
77      Addr.netId.b[1]=91;
78      Addr.netId.b[2]=120;
79      Addr.netId.b[3]=189;
80      Addr.netId.b[4]=1;
81      Addr.netId.b[5]=1;
82
83      // Fetch handle for the PLC variable
84      nErr = AdsSyncReadWriteReq(pAddr, ADSIGRP_SYM_HNDBYNAME
            ,
85                  0x0, sizeof(lHdlVar), &lHdlVar, sizeof(
                        szVar), szVar);
86      if (nErr) cerr << "Error: " << szVar <<
87                  " not found or no Connection to PLC " <<
                        nErr << '\n';
88      //
        ------------------------------------------------------------------------

89      cout << "Kommunikation zu Beckhoff gestartet\n";
90
91      valid_result [1][1]= 0 ;
92      for (i=1;i<L_VALID_RESULT;i++){
93          valid_result [i][0]=valid_result [i-1][0]+ANGLE_RES
                ;
94      }
95
96      while (1)
97      {
98          rc = recv(sConnect, recvbuf, size_buff, 0);
99          if (rc == SOCKET_ERROR)
```

```
100             std::cout << "error recv(): " <<
                    WSAGetLastError() << endl;
101         else
102         {
103             if (rc > int(3000) && rc < int(6000)  ){
104                 rcv = recv_meas(valid_result,r_last,
                        theta_last,v_con);
105             }
106             memset(&recvbuf, 0, sizeof(recvbuf));
107         }
108     }
109     closesocket(sConnect);
110     WSACleanup();
111     return 0;
112 }
```

Bibliography

Babaghasabha, R., Khosravi, M. A., and Taghirad, H. D. (2014). Vision based PID control on a planar cable robot. In *2014 22nd Iranian Conference on Electrical Engineering (ICEE)*, pages 1248–1253.

Barnett, V. (1978). The study of outliers: Purpose and model. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 27(3):pp. 242–250.

Beckhoff (2015a). Product documentation el6601 and el6614. `http://download.beckhoff.com/download/document/io/ethercat-terminals/el6601_el6614en.chm`,. [Accessed July 7, 2015].

Beckhoff (2015b). Twincat 3 system node. `http://infosys.beckhoff.de/english.php?content=../content/1033/tc3_system/html/tcsysmgr_systemnode_subnodes.htm&id=14266`. [Accessed July 13, 2015].

Beckhoff (2015c). Twincat 3 user interface. `http://infosys.beckhoff.com/index_en.htm`. [Accessed July 13, 2015].

Ben-Gal, I. (2005). *Data Mining and Knowledge Discovery Handbook*, chapter Outlier detection. Kluwer Academic Publishers, Netherlands.

Bruckmann, T. (2010). *Auslegung und Betrieb redundanter paralleler Seilroboter*. Ph. D. dissertation, Universität Duisburg-Essen, Duisburg, Germany.

Bruckmann, T., Mikelsons, L., Brandt, T., Hiller, M., and Schramm, D. (2008). Wire robots part I - kinematics, analysis & design. In Lazinica, A., editor, *Parallel Manipulators - New Developments*, ARS Robotic Books, pages 109–132. I-Tech Education and Publishing, Vienna, Austria. ISBN 978-3-902613-20-2.

Bruckmann, T., Sturm, C., Fehlberg, L., and Reichert, C. (2013). An energy-efficient wire-based storage and retrieval system. In *2013 IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, pages 631–636.

Bruckmann, T., Sturm, C., and Lalo, W. (2011). Wire robot suspension systems for wind tunnels. *Wind Tunnels and Experimental Fluid Dynamics Research*, pages 137–143.

Chellal, R., Cuvillon, L., and Laroche, E. (2015). *A Kinematic Vision-Based Position Control of a 6-DoF Cable-Driven Parallel Robot*, pages 213–225. Springer International Publishing, Cham.

Cho, S. H. and Hong, S. (2010). Map based indoor robot navigation and localization using laser range finder. In *Control Automation Robotics Vision (ICARCV), 2010 11th International Conference on*, pages 1559–1564.

Cisco (2016). Understanding jitter in packet voice networks. http://www.cisco.com/c/en/us/support/docs/voice/voice-quality/18902-jitter-packet-voice.html. [Accessed August 30, 2016].

Dallej, T., Gouttefarde, M., Andreff, N., Dahmouche, R., and Martinet, P. (2012). Vision-based modeling and control of large-dimension cable-driven parallel robots. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1581–1586.

Deutschman (2013). Instruction manual protocol developer. `http://www.deutschmann.de/en/products/download.php?file_id=498`. [Accessed July 4, 2015].

Deutschman (2014). Instruction manual universal fieldbus-gateway unigate cl ethercat. `http://www.deutschmann.de/en/products/download.php?file_id=560`. [Accessed July 5, 2015].

dit Sandretto, J. A., Daney, D., and Gouttefarde, M. (2013). *Calibration of a Fully-Constrained Parallel Cable-Driven Robot*, pages 77–84. Springer Vienna, Vienna.

Douglas, D. H. and Peucker, T. K. (1973). Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *Cartographica: The International Journal for Geographic Information and Geovisualization*, 10(2):112–122.

Duda, R. O. and Hart, P. E. (1972). Use of the Hough transformation to detect lines and curves in pictures. *Commun. ACM*, 15(1):11–15.

Emerald-MM (2003). Emerald-MM 4-channel multi-protocol serial port PC/104 module. User Manual.

Fang, S. (2005). *Design, Modeling and Motion Control of Tendon-based Parallel Manipulators*. Ph. D. dissertation, Gerhard-Mercator-University, Duisburg, Germany. Fortschritt-Berichte VDI, Reihe 8, Nr. 1076, Duesseldorf.

Fischler, M. A. and Bolles, R. C. (1981). Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381–395.

Giesler, B., Graf, R., Dillmann, R., and Weiman, C. (1998). Fast mapping using the log-Hough transformation. In *Intelligent Robots and Systems, 1998. Proceedings., 1998 IEEE/RSJ International Conference on*, volume 3, pages 1702–1707 vol.3.

Hoeller, F., Konigs, A., and Schulz, D. (2014). Autonomous reconnaissance and surveillance in urban structures - eurathlon 2013. In *Autonomous Robot Systems and Competitions (ICARSC), 2014 IEEE International Conference on*, pages 223–228.

Hokuyo (2012). Scanning laser range finder UTM-30LX/LN specification. `http://www.hokuyo-aut.jp/02sensor/07scanner/download/pdf/UTM-30LX_spec_en.pdf`. [Accessed July 6, 2015].

Hough, P. (1962). Method and means for recognizing complex patterns.

Illingworth, J. and Kittler, J. (1988). A survey of the Hough transform. *Computer Vision, Graphics, and Image Processing*, 44(1):87 – 116.

Jiménez, F. and Naranjo, J. E. (2011). Improving the obstacle detection and identification algorithms of a laserscanner-based collision avoidance system. *Transportation Research Part C: Emerging Technologies*, 19(4):658 – 672.

Kneubühl, F. K. and Sigrist, M. W. (2005). Laser. `http://link.springer.com/book/10.1007%2F978-3-322-99688-6`.

Kontron (2005). Produkt information speedMOPSlcdpm. `http://www.fortecag.de/mediadb/5923897/5923898/PPM1M090.pdf`. [Accessed July 6, 2015].

Kraus, W., Schmidt, V., Pott, A., and Verl, A. (2012). Investigation on a planar cable-driven parallel robot. In *Robotics; Proceedings of ROBOTIK 2012; 7th German Conference on*, pages 1–6.

Lalo, W. (2013). *Ein Beitrag zur Entwicklung von Assistenzsystemen für serielle and paralelle Roboter am Beispiel von Autobetonpumpen und seilbasierten Regalbediengeräten.* Ph. D. dissertation, Universität Duisburg-Essen, Duisburg, Germany.

Lee, K.-H. and Ehsani, R. (2008). Comparison of two 2D laser scanners for sensing object distances, shapes, and surface patterns. *Computers and Electronics in Agriculture*, 60(2):250 – 262.

Li, H., Lavin, M. A., and Master, R. J. L. (1986). Fast Hough transform: A hierarchical approach. *Computer Vision, Graphics, and Image Processing*, 36(2-3):139 – 161.

Li, H., Zhang, X., Yao, R., Sun, J., Pan, G., and Zhu, W. (2013). *Optimal Force Distribution Based on Slack Rope Model in the Incompletely Constrained Cable-Driven Parallel Mechanism of FAST Telescope*, pages 87–102. Springer Berlin Heidelberg, Berlin, Heidelberg.

Liu, Z., Liu, D., and Chen, T. (2013). Vehicle detection and tracking with 2d laser range finders. In *Image and Signal Processing (CISP), 2013 6th International Congress on*, volume 2, pages 1006–1013.

Merlet, J.-P. (2004). Solving the forward kinematics of a gough-type parallel manipulator with interval analysis. *International Journal of Robotics Research*, 23(3):221–236.

Merlet, J.-P. (2013). *Wire-driven Parallel Robot: Open Issues*, pages 3–10. Springer Vienna, Vienna.

Merlet, J.-P. and Alexandre-dit Sandretto, J. (2015). *The Forward Kinematics of Cable-Driven Parallel Robots with Sagging Cables*, pages 3–15. Springer International Publishing, Cham.

Microstrain (2015a). 3dm-gx1® Data Communication Protocol. `http://files.microstrain.com/manuals/3DM-GX1%20Data%20Communication%20Protocol%203102.pdf`. [Accessed October 6, 2015].

Microstrain (2015b). 3dm-gx1® timer ticks, calculation cycle and data output rates. `http://files.microstrain.com/tech_notes/TechNote_3DM-GX1_Timer_Tick_Intervals.pdf`. [Accessed October 6, 2015].

Miermeister, P., Kraus, W., Lan, T., and Pott, A. (2015). *An Elastic Cable Model for Cable-Driven Parallel Robots Including Hysteresis Effects*, pages 17–28. Springer International Publishing, Cham.

Ming, A. and Higuchi, T. (1994). Study on multiple degree-of-freedom positioning mechanism using wires (part 1). *International Journal of the Japan Society for Precision Engineering*, 28:131–138.

Mustang (2016). Mustang evolution automated storage/retrieval system. `http://login.tgw-group.com/at-en/products/asrs-solutions/mini-load-stacker-cranes/mustang-evolution-5/`. [Accessed July 29, 2016].

Nguyen, V., Martinelli, A., Tomatis, N., and Siegwart, R. (2005). A comparison of line extraction algorithms using 2D laser rangefinder for indoor mobile robotics. In *Intelligent Robots and Systems, 2005. (IROS 2005). 2005 IEEE/RSJ International Conference on*, pages 1929–1934.

Nocedal, J. and Wright, S. J. (2006). *Numerical Optimization*. Springer Verlag, 2 edition.

Pandilov, Z. and Dukovski, V. (2014). Comparison of the characteristics between serial and parallel manipulator. http://acta.fih.upt.ro/pdf/2014-1/ACTA-2014-1-19.pdf. [Accessed July 22, 2016].

Pott, A. and Schmidt, V. (2015). On the forward kinematics of cable-driven parallel robots. In *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, pages 3182–3187.

Qiu, Q. and Han, J. (2009). An implementation of typical-obstacle detection and recognition with laser range finder. In *Intelligent Computing and Intelligent Systems, 2009. ICIS 2009. IEEE International Conference on*, volume 2, pages 742–746.

Ramer, U. (1972). An iterative procedure for the polygonal approximation of plane curves. *Computer Graphics and Image Processing*, 1(3):244 – 256.

Reichert, C., Glogowski, P., and Bruckmann, T. (2015a). Dynamische rekonfiguration eines seilbasierten manipulators zur verbesserung der mechanischen steifigkeit. In *Tagung VDI Mechatronik 2015*.

Reichert, C., Unterberg, U., and Bruckmann, T. (2015b). Energie-optimale trajektorien für seilbasierte manipulatoren unter verwendung von passiven elementen. In *the 1st IFToMM D-A-CH Conference*, volume 1.

Saito, H., Amano, R., Moriyama, N., Kobayashi, K., and Watanabe, K. (2013). Emergency obstacle avoidance module for mobile robots using a laser range finder. In *SICE Annual Conference (SICE), 2013 Proceedings of*, pages 348–353.

Salah., B. (2013). *The implementation and analysis of a tendon based Stewart Gough platform (SGP) for an automated storage and retrieval system for mini-load*. Ph. D. dissertation, Universität Duisburg-Essen, Duisburg, Germany.

Salah, B., Ramadan, M., and Noche, B. (2011). Travel time analysis of stewart-gough platform in automated storage and retrieval system. In *Automation and Logistics (ICAL), 2011 IEEE International Conference on*, pages 143 –148.

Sanz-Cortiella, R., Llorens-Calveras, J., Escolà, A., Arnó-Satorra, J., Ribes-Dasi, M., Masip-Vilalta, J., Camp, F., Gràcia-Aguilá, F., Solanelles-Batlle, F., Planas-DeMartí, S., Pallejà-Cabré, T., Palacin-Roca, J., Gregorio-Lopez, E., Del-Moral-Martínez, I., and Rosell-Polo, J. R. (2011). Innovative lidar 3D dynamic measurement system to estimate fruit-tree leaf area. *Sensors*, 11(6):5769–5791.

Siadat, A., Kaske, A., Klausmann, S., Dufaut, M., and Hussen, R. (1997). An optimized segementation method for a 2d laser-scaner applied to mobile robot navigation. In *The 3rd IFAC Symposium on Intelligent Components and Instruments for Control Application*.

Sick (2015a). Industry guide. www.sick.com/media/pdf/9/79/879/IM0036879.PDF. [Accessed July 3, 2015].

Sick (2015b). Operating instruction LMS5xx laser measurement sensors. https://www.sick.com/media/pdf/4/14/514/IM0037514.PDF. [Accessed July 4, 2015].

Sick (2015c). Produktinformation sick LMS5xx. www.sick.com/media/pdf/5/65/165/IM0038165.PDF. [Accessed July 13, 2015].

Spong, M., Hutchinson, S., and Vidyasagar, M. (2006). *Robot modeling and control*. John Wiley & Sons.

Svoboda, T. (2008). Lecture note of RANdom SAmple Consensus. https://cw.fel.cvut.cz/wiki/_media/courses/a3m33iro/ransac.pdf. [Accessed August 6, 2015].

Verhoeven, R. (2004). *Analysis of the Workspace of Tendon-based Stewart Platforms*. Ph. D. dissertation, Universität Duisburg-Essen, Duisburg, Germany.

Wender, S., Weiss, T., and Dietmayer, K. (2005). Improved object classification of laserscanner measurements at intersections using precise high level maps. In *Intelligent Transportation Systems, 2005. Proceedings. 2005 IEEE*, pages 756–761.

Zuliani, M. (2014). Ransac for dummies. http://old.vision.ece.ucsb.edu/~zuliani/Research/RANSAC/docs/RANSAC4Dummies.pdf. [Accessed August 6, 2015].